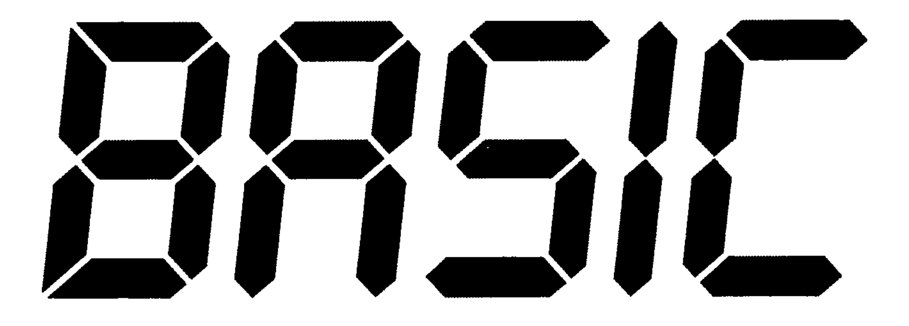
TAMIO SHIMIZU

BASIC

- Exercícios e problemas resolvidos
- Aplicações comerciais e científicas
- Simulação, jogos e gráficos
- Basic no ensino



atlas

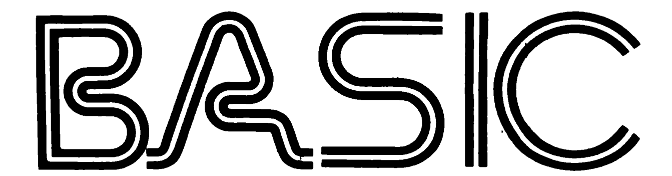


- Exercícios e problemas resolvidos
- Aplicações comerciais e científicas
- Simulação, jogos e gráficos
- Basic no ensino

atlas

BASIC

Tamio Shimizu



- Exercícios e problemas resolvidos
- Aplicações comerciais e científicas
- Simulação, jogos e gráficos
- Basic no ensino

SÃO PAULO
EDITORA ATLAS S.A. – 1984

Rua Helvetia, 574
Caixa Postal 7186 – Tel: (011) 221-9144
01215 São Paulo – SP

ISBN 85-224-0012-1

Impresso no Brasil/ *Printed in Brazil*

Diagramação
Pavel Gerencer

Capa
Paulo Ferreira Leite

1.ª EDIÇÃO — 1984

CIP-Brasil. Catalogação-na-Publicação
Câmara Brasileira do Livro, SP

S559b Shimizu, Tamio, 1938-
BASIC : exercícios e problemas resolvidos, aplicações comerciais e científicas, simulação jogos e gráficos, Basic no ensino / Tamio Shimizu. -- São Paulo : Atlas, 1984.

ISBN 85-224-0012-1

1. BASIC (Linguagem de programação para computadores) 2. BASIC (Linguagem de programação para computadores) - Problemas, exercícios, etc. I. Título.

83-2054

17. CDD-651.8
18. -001.6424
17. -651.8076
18. -001.6424076

Índices para catálogo sistemático:

1. BASIC : Linguagem de programação : Computadores : Processamento de dados 651.8 (17.) 001.6424 (18.)
2. Exercícios : BASIC : Linguagem de programação : Computadores :Processamento de dados 651.8076 (17.) 001.6424076 (18.)
3. Problemas : BASIC : Linguagem de programação : Computadores : Processamento de dados 651.8076 (17.) 001.6424076 (18.)

Sumário

Prefácio

Por que usar o Basic?

> "O Basic é uma linguagem de estrutura simples e de fácil aprendizado."

A afirmação acima deve ser literalmente obedecida de modo que um número cada vez maior de pessoas possa usufruir das vantagens educacionais dessa linguagem.

O Basic deve ser estudado com a prática de programação, através de exercícios, pois sua parte teórica é e deve permanecer simples.

Além disso, por sua simplicidade, o custo de aquisição e implantação do programa tradutor (chamado interpretador Basic) é baixo, ocupando espaço reduzido na memória, facilitando o seu uso em microcomputadores.

O Basic (Beginner's All-purpose Symbolic Instruction Code) foi desenvolvido no Darthmouth College, U.S.A., para ajudar os estudantes no treinamento de programação, usando terminal remoto, de modo conversacional em sistemas denominados *time-sharing* ou tempo partilhado.

Com o impulso do uso de micromputadores pessoais, o Basic ganhou o seu destaque, pois apresenta vantagens em atender estudantes, profissionais e *hobbystas* de diversos níveis e áreas, por ser uma linguagem simples e relativamente eficiente.

O Basic possui as características básicas para atender às necessidades e gostos de qualquer pessoa que vê no microcomputador um instrumento eficiente de cálculo.

As características principais do Basic são:

- operação simples e de caráter conversacional, com a execução imediata dos comandos e programas;
- facilidade em definir ou usar:
  - *nome de variáveis (uma letra ou letra e algarismo do tipo* A, B, C, X1, X2, A3 etc.);
  - tipo de variáveis (em geral todos os valores numéricos são reais);

- tipo de comandos (**LET, INPUT, PRINT, FOR'** etc. são de fácil entendimento);
- estruturas simples (as sub-rotinas **GOSUB** estão dentro do próprio programa);
- variáveis sem formato (dispensam o uso obrigatório do **FORMAT** como ocorre no FORTRAN).

Entretanto, algumas expansões naturais das características básicas do BASIC, tais como:

- edição de dados através do comando **PRINT USING**
- nome de variáveis com mais de dois caracteres
- valores com dupla precisão
- operação com *strings* de caracteres,

foram exigências naturais dos usuários ávidos em resolver problemas mais sofisticados.

Cabe ressaltar que não se deve recorrer a versões muito sofisticadas do BASIC, a tal ponto de se ter uma linguagem mais difícil de se aprender do que o FORTRAN, COBOL, PL/1, ou PASCAL, pois o Basic não possui a estruturação suficiente para se tornar complexa, gerando dificuldades no seu aprendizado e padronização da linguagem. Em tais casos seria preferível recorrer a linguagens com estruturas mais apropriadas como o PASCAL ou COBOL.

O BASIC deve ser usado por pessoas que se iniciam no campo da computação e processamento de dados, através de seu microcomputador simples e barato.

É essencial apresentar uma visão progressiva e simples do uso de seus comandos (sempre através de exercícios e exemplos) sem entrar em detalhes especiais das diferentes versões existentes no mercado. Tais detalhes podem ser examinados por cada interessado após obter a noção básica do BASIC.

O lema principal deste texto é apresentar o BASIC de modo a:

"Motivar e ensinar pessoas novas no ramo a resolver seus problemas usando um microcomputador."

O texto está dividido em duas partes: A primeira contém apresentação e exercícios sobre o uso dos principais comandos BASIC, e a segunda contém aplicações em diversos ramos da atividade.

Toda a tentativa de oferecer exemplos e programas BASIC testados foi feita pelo autor. Entretanto, é quase impossível executar testes exaustivos, e eliminar totalmente erros de transcrição e adaptações. Sendo assim estamos prontos a receber e agradecemos antecipadamente qualquer notificação sobre os eventuais erros encontrados.

Registramos os sinceros agradecimentos à colaboração prestada pelo colega, Sr. Mário Sobolewski, da FEI-IPEI, no trabalho de pesquisa e levantamento das diferentes formas dos comandos e funções existentes no Basic.

O Autor

O Basic e o seu Microcomputador 1

1.1 PROCESSAMENTO DE DADOS COM MICROCOMPUTADORES

O uso de microcomputadores, tanto para serviços pessoais como para processamento de dados em pequenas empresas, tem crescido bastante devido ao baixo custo desses equipamentos e eficiência cada vez maior de *software*. (Veja Fig. 1.1.)

As facilidades oferecidas por uma linguagem de caráter simples e conversacional denominada Basic, tem contribuído bastante para o maior interesse dos usuários em resolver problemas científicos e comerciais.

1.2 DEFINIÇÃO DE ALGUNS TERMOS TÉCNICOS USADOS NO TEXTO

Os seguintes termos técnicos sobre computadores podem aparecer no texto:

ARQUIVO DE DADOS – Conjunto de dados agrupados sob a mesma forma física (cartão, fita magnética, fichas) para determinada finalidade.

ASCII – (American Standard Code for Information Interchange) – Sistema de representação binária de caracteres alfanuméricos padronizado para uso em sistema de informação e computadores. (Ver Apêndice B.)

BINÁRIO – Sistema de numeração que usa *bits* com valor 1 ou 0.

COMPILADOR – Programa que traduz programas em linguagem de alto nível (FORTRAN, COBOL, PL/I) para linguagem de máquina. O Basic pode possuir o seu programa compilador embora seja freqüente usar o interpretador Basic.

CONVERSACIONAL – Característica da operação ou linguagem que permite ao usuário escrever uma instrução e receber imediatamente o resultado, sem demora.

DISCO MAGNÉTICO — Meio ou arquivo de armazenamento de dados de grande capacidade e que opera no modo direto ou randômico, podendo buscar os dados diretamente em qualquer parte do disco sem obedecer seqüências.

DISKET ou DIQUETE — Disco magnético de porte pequeno, próprio para microcomputadores. É feito de material plástico flexível e também conhecido por FLOPPY-DISK.

DISPLAY ou TELA — Meio de representação visual, normalmente feito através de vídeos de TV ou CRT (Cathode Ray Tube).

DRIVER — Unidade que recebe e opera os meios de entrada/saída como disquete, fita etc.

FITA MAGNÉTICA — Meio ou arquivo de armazenamento de dados de capacidade média e que opera de modo seqüencial, lendo ou gravando dados sempre do começo ao fim da fita.

FLUXOGRAMA — Notação gráfica que representa o raciocínio lógico para execução de um programa.

HARDWARE — Equipamentos ou peças físicas (ferramental) do computador.

INTERPRETADOR — Programa que executa cada comando de outro programa à medida que o encontra. Diferente do Compilador, que traduz o programa todo executando-o depois.

KEY-BOARD — Ver TECLADO.

MEMÓRIA — Parte do computador onde os dados (programas e valores) são armazenados. Pode ser formada de núcleo magnético de ferrite ou por peças de circuitos integrados chamadas *chips*.

MEMÓRIA RAM — (RANDOM ACCESS MEMORY) — Memória de circuito integrado onde é possível ler e guardar qualquer dado ou programa.

MEMÓRIA ROM — (READ ONLY MEMORY) — Memória de circuito integrado de onde só é possível ler dados, não podendo efetuar o armazenamento.

MICROCOMPUTADOR — Computador de porte pequeno que utiliza na sua parte central ou principal de processamento, o microprocessador.

MICROPROCESSADOR — Peça ou pastilha ou *chip* de circuito integrado que contém todas as funções principais da unidade central ou principal de processamento (UCP) de um computador.

PLOTTER — Unidade de saída para traçar gráficos.

SOFTWARE — Conjunto de programas básicos de sistema (como o Interpretador Basic) e programas aplicativos de um computador.

TECLA OU TECLADO — (ver KEY-BOARD) — Parte de um terminal por onde são datilografados ou digitados os códigos, valores e certas operações (como mudar de linha) de um programa.

TERMINAL – Equipamento de entrada ou saída de dados colocado para possibilitar o acesso de dados (programa ou valores) do usuário ao computador e vice-versa. Pode ser terminal de teletipo, teclado alfanumérico, vídeo, impressora etc.

UCP – (Unidade Central de Processamento). Parte principal ou central de computador e que executa as operações aritméticas, lógicas, de transferência de dados etc.

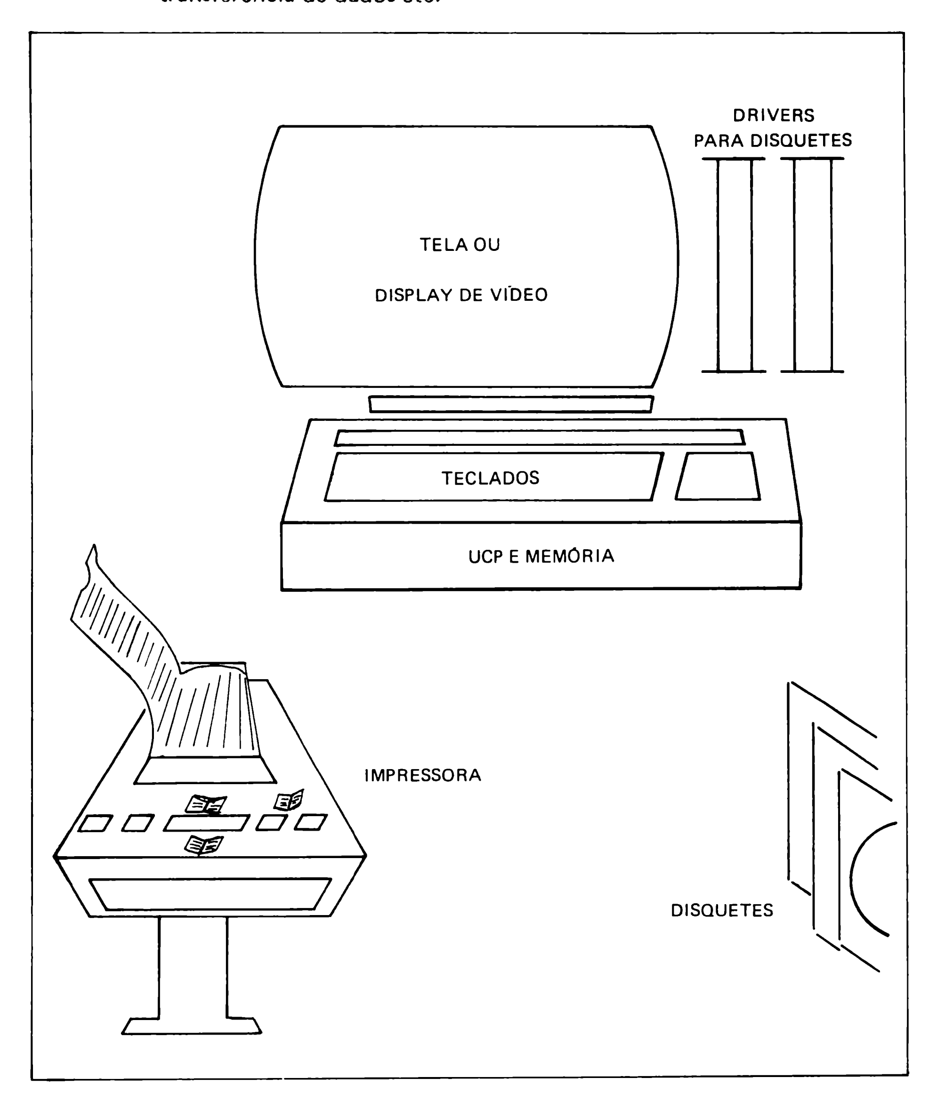
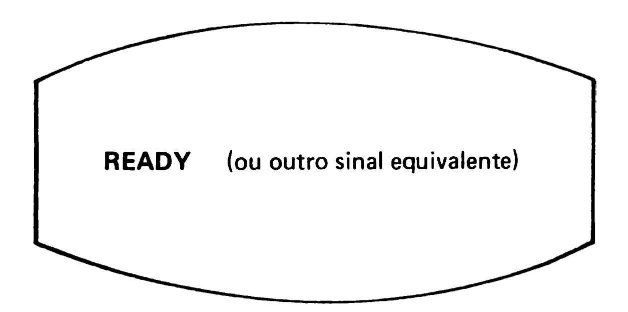


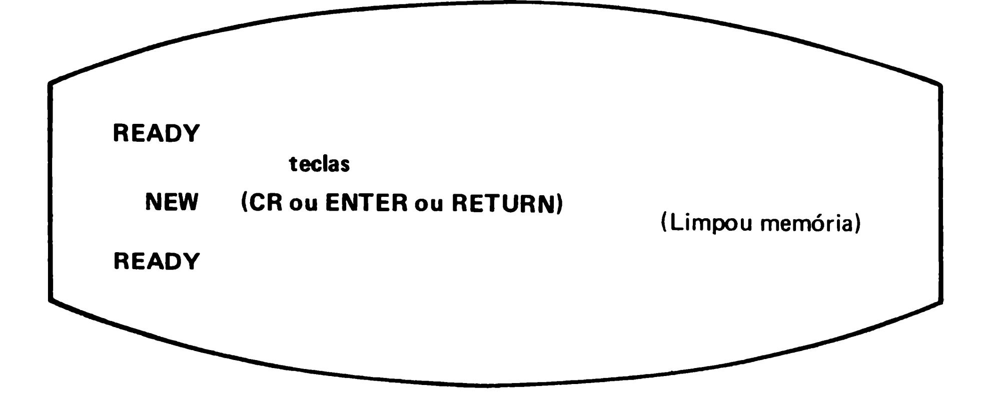
Figura 1.1 *Configuração de um Microcomputador*

1.3 PROCESSAMENTO DE UM PROGRAMA EM BASIC: OPERAÇÕES BÁSICAS DO TECLADO

- O terminal do computador, usando Basic, estará pronto para receber um programa. Na tela do terminal (ou na impressora) aparecem os seguintes sinais:



- Escreve-se o termo **NEW** no terminal, sempre que o programa for novo; caso contrário, o mesmo será acrescentado ao programa já existente na memória.
- Após o **NEW,** apertar a tecla **RETURN** (ou **ENTER** ou **CR** – Carriage Return). O computador escreverá a mensagem **READY** e o sinal # ou]. (Figura 1.2)



- Escreve-se apenas um comando do programa de cada vez, com o respectivo número, e aperta-se a tecla de "Retorno de Linha" do terminal, o que coloca o comando na memória (tecla **CR**-Carriage Return, Return ou Enter).

- Os comandos podem entrar fora da seqüência numérica, pois o Basic coloca-os na ordem correta. Isto permite a inserção de novos comandos no meio do programa, se existir número de comandos vagos.
- Um novo comando com número de comando já usado apaga o comando anterior da memória.

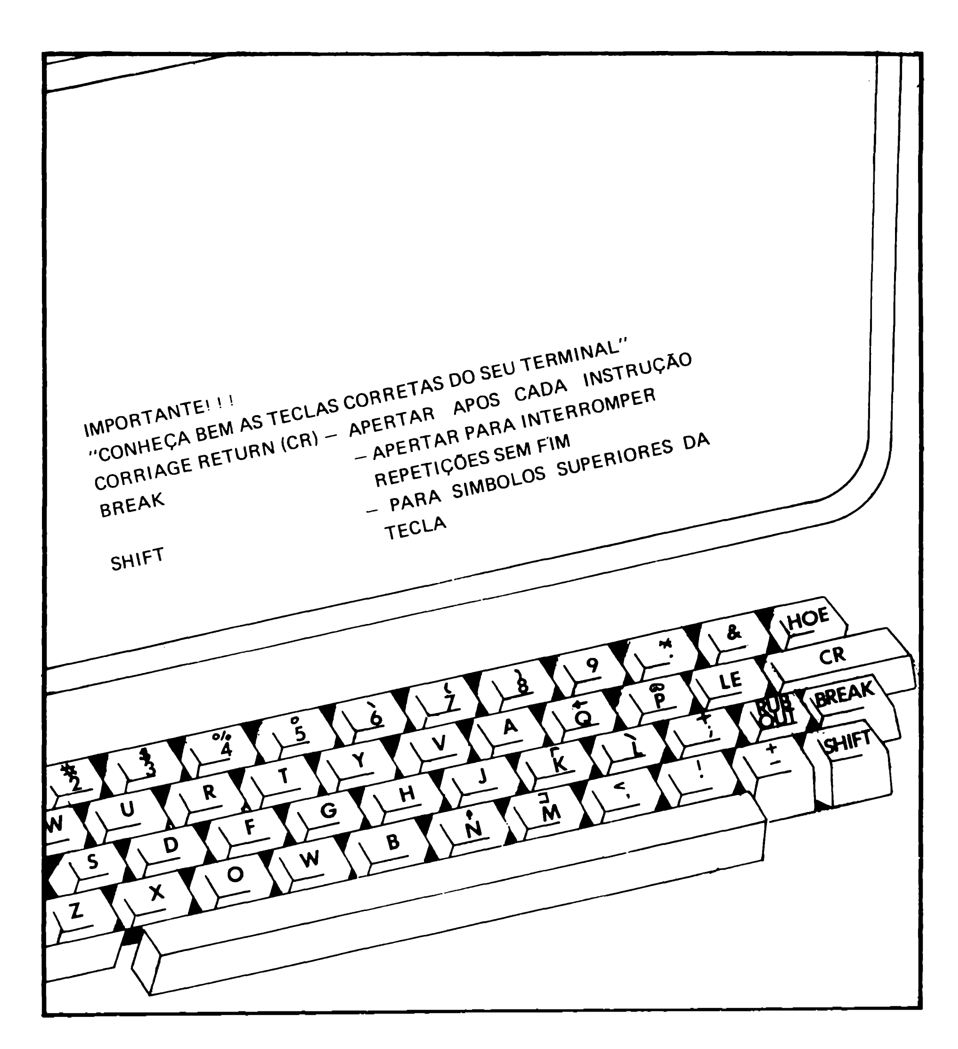


Figura 1.2. – *É preciso conhecer as teclas do seus sistema*

1.3.1 Exemplos demonstrativos de programa em Basic

Os seguintes programas simples podem ser datilografados, linha por linha, e executados pelo leitor. Os programas dispensam maiores explicações.

Após datilografar uma linha, o leitor deve sempre apertar a tecla de RETORNO DE LINHA (CR, ENTER ou RETURN).

a) *Calcular valores*

```
NEW
10 PRINT 3+5+10, 3*5
20 END
RUN
    18      15 (Resultados)
```

Um programa equivalente seria:

```
NEW
10 LET A= 3+5+10
20 LET B= 3*5
30 PRINT A,B
40 END
RUN
    18      15 (Resultados)
```

b) *Escrever nome e endereço*

```
NEW
10 PRINT "SR ANTONIO UBALDINO REZENDE"
20 PRINT "RUA DOS PATOS, S/N"
30 PRINT "S.CARLOS – S.PAULO"
40 PRINT
50 END
RUN
    SR. ANTONIO UBALDINO REZENDE
    RUA DOS PATOS, S/N
    S.CARLOS – S.PAULO                (Resultados)
```

Datilografando a palavra RUN, o programa que está na memória será executado novamente. O leitor pode colocar o seu próprio nome e endereço, modificando os comandos 10, 20 e 30.

c) *Traçar um triângulo na tela*

```
NEW
10 PRINT "X"
20 PRINT "XX"
30 PRINT "XXX"
40 PRINT "XXXX"
50 PRINT "XXXXX"
60 END
RUN
X
XX
XXX
XXXX
XXXXX
```

Cada um dos comandos ou instruções destes programas serão estudados com detalhes e exemplos, a partir do próximo capítulo.

Na seção seguinte, estudaremos a *forma geral* de um programa em BASIC, ainda através de um exemplo simples, sem entrar em detalhes sobre cada comando.

1.4 A LINGUAGEM BASIC

O BASIC (Beginner's All-purpose Symbolic Instruction Code) é uma linguagem extremamente fácil, destinada à programação de aplicações científicas e aplicações comerciais simples.

Seu caráter conversacional (isto é, o programador pode executar e corrigir o seu programa diretamente no terminal) proporciona estrutura simples aos comandos, embora permita a programação de problemas bastante complexos.

Um exemplo de programa em Basic:

"Calcule a fórmula: X = (A+B) ÷ (3xC)."

Vamos tentar apenas entender o seguinte programa BASIC. Não é difícil.

Comandos	Explicação
NEW	• Prepara o computador para receber um programa novo
100 INPUT A,B,C	• Recebe os valores de A,B e C pelo terminal
101 LET X = (A+B) / (3 *C)	• Calcula a fórmula
105 PRINT X	• Escreve o valor de X pelo terminal
109 END	• Fim do programa
RUN	• Vai executar o programa lido.

Observações:

- O leitor não deve estranhar certas *convenções* adotadas pelo computador e que são diferentes do nosso raciocínio normal, tais como: ler valores A,B,C *antes* de conhecer a fórmula, ou, colocar sinal de igualdade *antes* da fórmula.
- Todo comando Basic é precedido pelo número do comando.
- O programa é executado em ordem crescente dos números do comando, exceto quando houver uma instrução de desvio, como **GOTO** ou **IF**.
- **NEW** e **RUN** são *Comandos de Controle* do programa e não fazem parte da linguagem Basic; por essa razão, não possuem o número de comando.
- Espaços em branco podem ser usados à vontade, mas cada comando não deve exceder 256 caracteres.
- Os números do comando podem variar de **0001** a **9999**.

```
NEW
READY
100 INPUT A,B,C (apertar tecla CR ou ENTER ou RETURN após
                 cada comando)
101 LET X = (A+B) / (3*C)            (coloca comandos
105 PRINT X                          BASIC na memória)
109 END
```

- Após terminar a colocação do programa na memória com o comando **END**, pode-se usar qualquer uma das alternativas seguintes:

  RUN – Para executar o programa.
  LIST – Para listar o programa.
  SAVE – Para guardar o programa na memória em forma de fita ou cartão.

- A qualquer instante podem-se adicionar novos comandos ao programa da memória, ou então apagá-lo com o comando **NEW**.

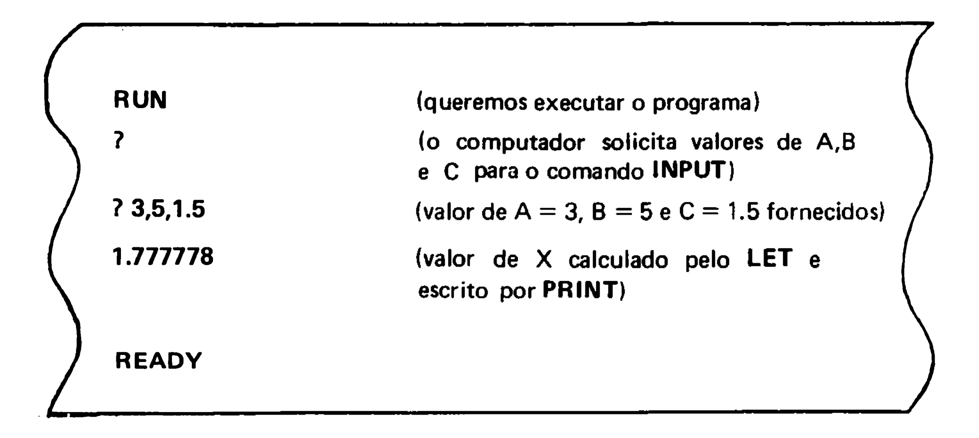
1.5 ALTERNATIVAS DE OPERAÇÃO

1.5.1 Opção RUN

Supondo que o programa já foi colocado na memória,

```
NEW
READY
100 INPUT A,B,C
101 LET X = (A+B) / (3*C)
105 PRINT X
109 END
```

datilografando o termo RUN, após o comando END, temos:



1.5.2 Opção **LIST**

Fornece nova listagem de todo o programa que está na memória.

```
LIST                                 (Listar programa BASIC
100 INPUT A,B,C                          da memória)
101 LET X = (A+B) / (3*C)
105 PRINT X
109 END
READY
```

1.6 COMANDOS DE CONTROLE DE PROGRAMA

São comandos que permitem o controle direto do programa, por parte do usuário com a utilização de diversas alternativas possíveis. Esses comandos, em geral, não possuem número de comando e têm prioridade de execução imediata, não sendo guardados na memória.

NEW Limpa a memória e prepara o computador para receber um novo programa.

RUN Executa o programa que está na memória, a partir do comando de número mais baixo e até encontrar um comando **STOP** ou **END.**

LIST Lista o programa ou trecho do programa que está na memória.

Exemplos:

LIST	(lista o programa inteiro)
LIST 100, 200	(lista o comando 100 a 200)
ou **LIST 100-200**	
LIST 100, 100	(lista o comando 100)
ou **LIST 100**	

SAVE Guarda o programa da memória em fita de papel ou cassete para posterior execução ou arquivamento.

LOAD Carrega o programa que foi guardado pelo comando **SAVE.**

APPEND Idêntico ao **LOAD,** só que permite que o programa da fita ou cartão seja acrescentado ao programa existente na memória.

Alguns comandos adicionais estão listados no Apêndice C.

1.7 O BASIC COMO MÁQUINA DE CALCULAR: EXECUÇÃO IMEDIATA DE COMANDOS SEM NÚMERO

Os comandos de controle **NEW, LIST, RUN** etc. são em geral usado *sem o número de comando,* e por isso, executados imediatamente, sem serem guardados na memória. Dizemos que a execução foi *direta ou imediata.*

Entretanto, qualquer outro comando Basic (como o **PRINT, LET** etc.) pode ser usado em modo direto sem ser precedido de um número de comando. Esse comando é imediatamente executado, sem ser armazenado na memória.

Exemplo:

NEW	(executado imediatamente)
READY	
PRINT 3, 7+8	(executado imediatamente)
3 15	(resultados do **PRINT**)
READY	
LET Z = 3+7	(a variável Z recebe imediatamente o valor 3+7 = 10)
READY	

1.8 EXECUÇÃO DE COMANDO ATRAVÉS DE RUN

Os comandos Basic precedidos de um número de comando são guardados na memória e executados, todos de uma só vez, através do comando **RUN**. Sofrem, portanto, uma execução indireta.

Entretanto, qualquer comando de controle (**NEW, LIST, SAVE** etc.) também pode ser executado de modo indireto, juntamente com o programa Basic.

Exemplo:

```
100 INPUT A,B,C
101 LET X = (A+B) / (3*C)
105 PRINT X
106 LIST 100, 105 (LIST usado de modo indireto)
109 END
```

Resultado:

```
RUN
? 3, 5, 1.5
 1.777778                    Execução pelo RUN
100 INPUT  A, B, C
101 LET X  =  (A+B) / (3 *C)  Listagem pelo LIST 100, 105
105 PRINT X
READY
```

Observações:

- Se usarmos o comando **NEW** no modo indireto, todo o programa será destruído, por ocasião da execução pelo **RUN**.
- O que acontece após a execução de **106 LIST** depende da versão BASIC, pois tanto pode voltar ao estado **READY** como continuar no estado de execução **RUN**. No exemplo anterior, o termo **READY** poderia ser produzido tanto pelo comando **LIST** como pelo comando **END**. Se utilizarmos **104 LIST 100, 101**, antes do comando **105 PRINT X** poderia acontecer um dos casos seguintes:

a) *Primeiro caso*

```
 RUN
 ? 3, 5, 1.5                (LIST sem retorno ao RUN)
100 INPUT A, B, C
101 LET X = (A+B) / (3 *C)
 READY
```

b) *Segundo caso*

```
RUN
? 3 , 5, 1.5               (LIST com retorno ao RUN
100 INPUT A, B, C          executando o PRINT)
101 LET X = (A+B) / (3 *C)
1.777778
READY
```

1.9 EXERCÍCIOS DE APLICAÇÃO

1. O que significa Basic?
2. Quais são as características de uma linguagem Basic?
3. Dizer o que fazem os seguintes comandos Basic

NEW, LIST, RUN, SAVE, LOAD

```
100 INPUT A, B, C
101 LET X = (A+B)
105 PRINT X
107 PRINT X, A, B
109 END
```

4. Definir e explicar os seguintes conceitos:
   - Terminal de microcomputador
   - Memória de microcomputador
   - Comando Basic
   - Comando de Controle

5. O que é modo direto e modo indireto de execução de comando Basic? Dar exemplos.

6. Quais as operações básicas do teclado para:
   a) Limpar memória?
   b) Carregar um programa Basic?
   c) Executar um programa?
   d) Listar a instrução **NN** do programa?

Comandos INPUT, LET, PRINT, REM e END 2

2.1 COMANDOS UTILIZADOS NO CAPÍTULO (Ver Fig. 2.1)

Passaremos a estudar o uso de cada comando BASIC.

INPUT	Recebe do terminal valores para as variáveis definidas no comando. *Exemplo:* **50 INPUT A, Z1, C** Ao chegar a este comando, o programa interrompe a execução imprimindo o sinal ?. O programador deve fornecer, pelo terminal, três valores numéricos separados por vírgula ou espaço.
LET	Executa as operações aritméticas. A palavra **LET** é opcional em muitas versões de Basic. *Exemplo:* **103 LET A = A + X** **105 A = A + X * Y** Operações possíveis: – adição (+) – divisão (/) – subtração (-) – potência (↑ ou **) – multiplicação (*)
PRINT	Escreve no terminal os valores das variáveis mencionadas no comando ou os comentários, colocados entre 2 sinais de aspas. *Exemplo:* **20 PRINT "VALORES", A, B, C** Se na memória existirem os valores A = 5, B = 1 e C = 10, este comando escreverá, no terminal, a linha. **VALORES 5 1 10**

END	Indica fim do programa Basic. Cada programa deveria usar um único comando **END**. Se houver necessidade de terminar a execução em mais de um ponto do programa, deve-se usar o comando **STOP**. Porém, na maioria das versões do Basic o uso de vários **END** no meio do programa é permitido.
REM	Usado como comentário, anotação ou título do programa. *Exemplo:* **111 REM CALCULO DA FORMULA X**

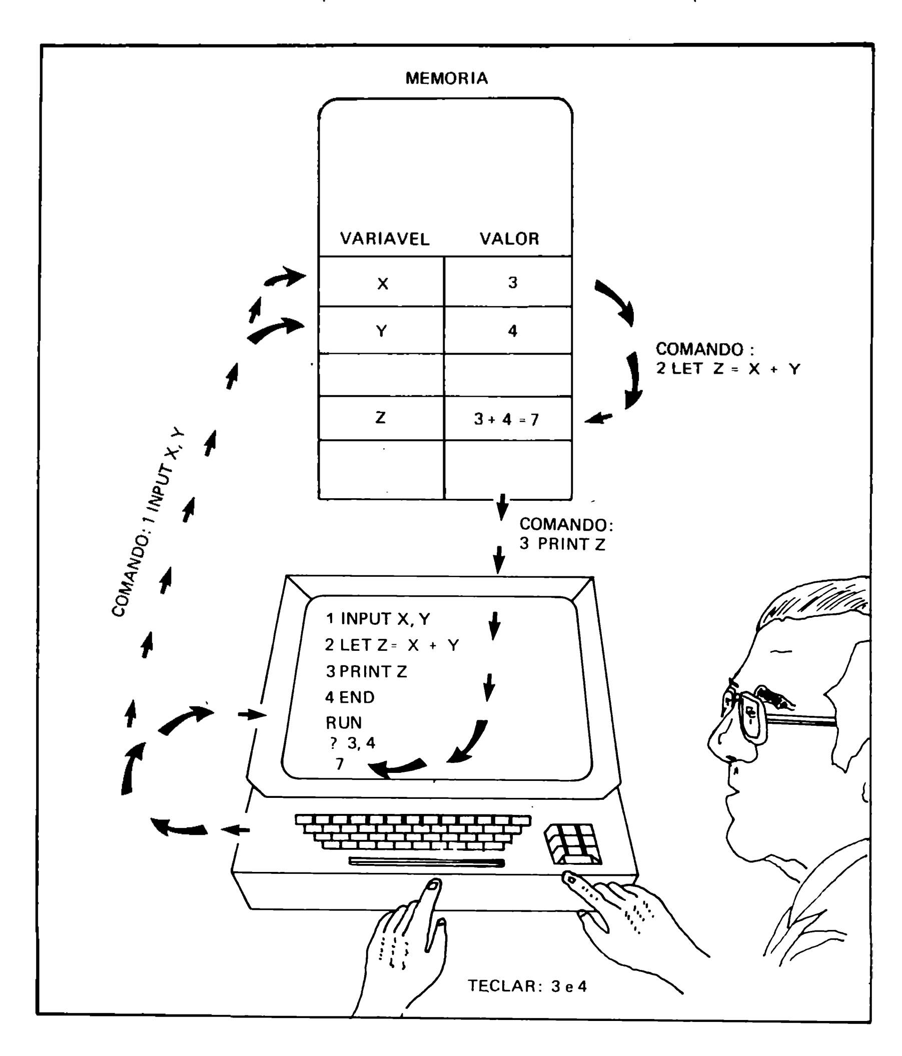


 Figura 2.1 – *Comandos INPUT, LET e PRINT*

EXERCÍCIOS DE LER, ESCREVER E CALCULAR

Escrever o comentário *"OBA OBA"* pelo computador. Comando utilizado: *PRINT "...", END, NEW* e *RUN.*

Programa:

```
NEW
10  PRINT "OBA-OBA"
20  END
RUN


 OBA OBA  (Resultado)
```

FLUXOGRAMA

ESCREVER: "OBA-OBA"

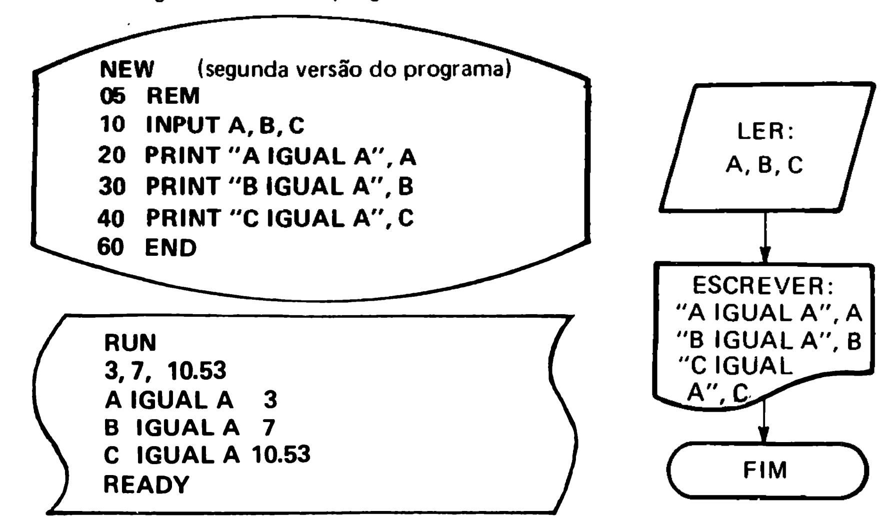
FIM

Ler e escrever os valores **A,B,C.**
Comandos utilizados: **INPUT, PRINT.**

```
NEW
10  INPUT A,B,C.
20  PRINT A,B,C.
40  END.
```

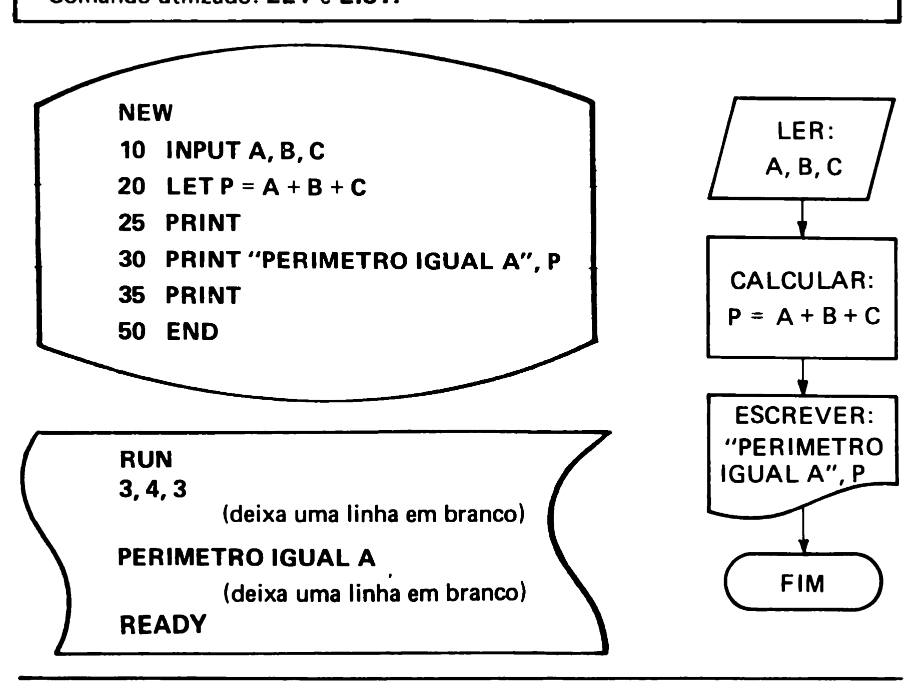
```
RUN
?    (comando INPUT solicitando valores de A, B e C)
3,7, 10.53  (valores teclados pelo usuário)
3  7  10.53  (valores impressos pelo cumputador)
READY
```

Uma segunda versão do programa com comentário seria:



Nota: Comando **PRINT** sem variável ou comentário, permite a operação "pular uma linha."

Calcular e escrever o perímetro do triângulo de lados A, B e C. Comando utilizado: **LET** e **LIST.**



LIST (vai listar o programa que está na memória)

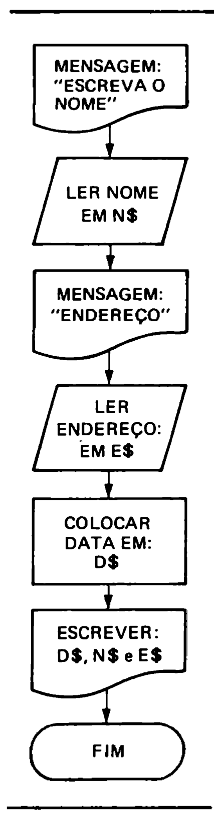
```
10 INPUT A,B,C
20 LET P=A+B+C
25 PRINT
30 PRINT "PERIMETRO IGUAL", P
35 PRINT
50 END
READY
```

Ler e escrever nome e endereço de uma pessoa: variáveis não numéricas ou *strings:*

Programa:

```
NEW
5 PRINT "ESCREVA O NOME"
10 INPUT N$
20 PRINT
30 PRINT "ENDEREÇO"
40 INPUT E$
45 LET D$ = "12/OUT/1989"
50 PRINT
60 PRINT "DATA, NOME E ENDEREÇO"
70 PRINT D$
80 PRINT N$
90 PRINT E$
100 END
```

FLUXOGRAMA:



Resultados:

```
RUN
ESCREVA O NOME
? JOAQUIM XAVIER        (nome fornecido)
ENDEREÇO
? RUA XODOH, N.13       (endereço)
DATA, NOME E ENDEREÇO
12/OUT/1989
JOAQUIM XAVIER
RUA XODOH, N.13
READY
```

2.2 VARIÁVEIS E CONSTANTES NUMÉRICAS

O Basic trabalha com *variável,* que assume qualquer valor dado; e com constante, que assume valor fixo. As variáveis podem ser de dois tipos: numéricas e não numéricas.

Todas as variáveis numéricas recebem valor zero ao iniciar o processamento do programa.

2.2.1 Variáveis numéricas simples

São definidas por uma letra, ou por letra seguida de dígito. Algumas versões permitem o uso de mais de dois caracteres.

Exemplos:

X, Y, Z1, A9

XX, 9A, A34 (formas erradas, mas aceitas em algumas versões)

2.2.2 Constantes

O Basic aceita valores numéricos entre $\pm 1{,}0 \times 10^{-99}$
$\pm 1{,}0 \times 10^{99}$, nas seguintes
formas:

Inteira, sem ponto decimal: **1, 99 , 10, 99999**

Decimal: **0.123 , 1.0, 99.978**

Forma Exponencial ou em ponto Flutuante

-1.234E 03 **+59 E-43** (que vale 59×10^{-43})

2.2.3 Variáveis estritamente inteiras

Muitas versões do Basic permitem distinguir variáveis com os valores inteiros, colocando o sinal de porcentagem (%) imediatamente após o nome da variável.

Exemplo:

X% I2% (só podem conter valores inteiros como -5, 0, 35 etc.)

2.2.4 Variáveis com nomes longos (mais de uma letra ou dígito)

Algumas versões mais poderosas do Basic (e que gastam mais memória) permitem o uso de nomes de variáveis mais longas, possibilitando melhorar o significado dado a essas variáveis.

Exemplo:

DATA, NOME, ROTULO-1

2.3 VARIÁVEIS E CONSTANTES NÃO NUMÉRICAS OU STRINGS

Quando o nome de uma variável simples estiver seguida pelo sinal ($) ela receberá apenas valores ou constantes não numéricos, isto é, caráter alfabético ou seqüência de caracteres alfabéticos denominadas *strings*. Estas variáveis não podem ser utilizadas em operações aritméticas, mas são usadas em operação de Entrada ou Saída, Comparação, e de designação simples de valores.

Exemplos:

A$, B$, X$	(variáveis não numéricas)
"ZERO", "JOSE SILVA", "015"	(constantes alfanuméricas)
LET A$ = "ANTONIO"	(variável A$ contém ANTONIO)
PRINT A$	

2.4 EXPRESSÕES ARITMÉTICAS

O Basic permite as seguintes operações aritméticas:

↑ ou **	potenciação
*	multiplicação
/	divisão
+	adição
-	subtração

Exemplo:

Expressão	Basic
$\frac{(x^2 + y)}{3}$	(X ↑ 2 + Y)/3
$2 \pi R^2$	2 *3.1416 *R ↑ 2

Calcular os valores das seguintes expressões aritméticas:

$$X = A + B^3$$

$$Y = A + \frac{B}{C + D}$$

$$Z = \frac{A + B}{C + D} - A.B$$

$$W = \frac{C \times A}{B}$$

```
NEW
10 INPUT A,B,C,D
20 LET X = A + B ↑ 3              (Potenciação)
30 LET Y = A + B/(C+D)            (Parêntese obrigatório)
40 LET Z = (A+B)/(C+D) - A *B     (Idem)
50 LET W = C *A/B                 (Parêntese desnecessário)
60 PRINT X, Y, Z, W
80 END
```

Regra de Precedência ou Prioridade nas operações aritméticas.

Em geral, a regra de precedência, ou prioridade obedecida para a execução das operações aritméticas é a seguinte:

a) Dentro do mesmo nível ou conjunto de parênteses:

Prioridade 1 : Potenciação
2 : Multiplicação e Divisão
3 : Adição e Subtração

b) Para duas operações de mesmo nível e mesma prioridade, como Adição e Subtração, a prioridade é da esquerda para a direita, isto é, a operação que está mais à esquerda é executada em primeiro lugar.

c) As prioridades anteriores são sempre suspensas cada vez que for encontrada uma operação "Entre Parênteses", que possui novo nível ou prioridade mais alta e deve ser executada primeiro.

Portanto:

LET A = X/(Y-D) é diferente de **LET A = X/Y-D**

LET A = (C+D)/(B+W) é diferente de **LET A = (C+D)/B+W** ou
LET A = C+D/(B+W) ou
LET A = C+D/B+W

Exemplo: para C = 2, B = 3, D = 5 e W = 3

LET A = (C + D) / (B + W) = (2 + 5) / (3 + 3) = 7/6
LET A = (C + D) / B + W = ((2 + 5) / 3) + 3 = 7/3 + 3
LET A = C + D /(B + W) = 2 + (5/(3 + 3)) = 2 + 5/6
LET A = C + D/B + W = 2 +(5/3) + 3 = 5/3 + 5

2.5. APLICAÇÃO COMERCIAL: LISTAGEM DE RELATÓRIO

Programa para calcular e listar resumo de PNB (Produto Nacional Bruto)

Escrever um programa em Basic que efetue as seguintes operações:

Ler valores de:	Produção Industrial	D = 179.3 (em bilhões de cruzeiros)
	Produção Agropastoril	N = 251.5
	Serviços	S = 201.7
	Compras do Governo	G = 201.5
	Investimento no País	N1 = 53.0
	Exportações	E = 50.6
	Importações	I = 41.3
	Depreciações	D1 = 53.3

Calcular os valores de:

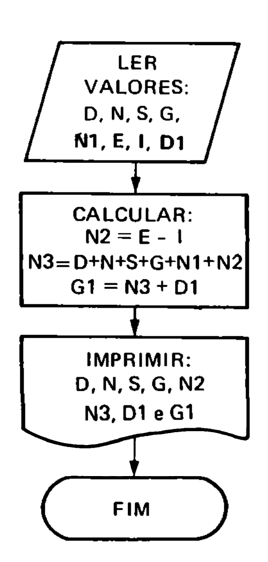
Balanço do Comércio Exterior: N2 = E - I

Produto Nacional Líquido: N3 = D + N + S + G+N1 + N2

Produto Nacional Bruto: G1 = N3 + D1

Escrever uma tabela dos valores lidos, do produto nacional líquido e bruto.

FLUXOGRAMA



```
0100 PRINT "FORNECER VALORES DE D,N,S,G"
0115 INPUT  D,N,S,G
0118 PRINT "FORNECER VALORES DE N1, E, I, D1"
0120 INPUT N1,E,I,D1
0150 PRINT
0160 PRINT
0170 PRINT "b̸", "CALCULO DO PRODUTO NACIONAL"
0180 PRINT "b̸", "b̸", "LIQUIDO E BRUTO"
0190 LET N2=E-I
0200 LET N3=D+N+S+G+N1+N2
0210 LET G1=N3+D1
0220 PRINT
0230 PRINT "===================================="
0240 PRINT "PRODUCAO INDUSTRIAL                 ",D
0250 PRINT "PRODUCAO AGRO-PASTORIL              ",N
0260 PRINT "SERVICOS                            ",S
0270 PRINT "COMPRA DO GOVERNO                   ",G
0280 PRINT "INVESTIMENTOS NO PAIS               ",N1
0290 PRINT "BALANCO DO COMERCIO EXTERIOR        ",N2
0300 PRINT "                                    ","........."
0310 PRINT "PRODUTO NACIONAL LIQUIDO            ",N3
0320 PRINT "DEPRECIACAO                         ",D1
0330 PRINT "                                    ","........."
0340 PRINT "PRODUTO NACIONAL BRUTO              ",G1
0350 PRINT "===================================="
0360 END
```

```
#RUN
FORNECER VALORES DE D,N,S,G
? 179.3,261.5,201.7,201.5
FORNECER VALORES DE N1,E,I,D1
? 53.0,50.6,41.3,53.3

   CALCULO DO PRODUTO NACIONAL
   LIQUIDO E BRUTO
========================================
PRODUCAO INDUSTRIAL                179.3
PRODUCAO AGRO-PASTORIL             251.5
SERVIÇOS                           201.7
COMPRAS DO GOVERNO                 201.5
INVESTIMENTOS NO PAIS              53
BALANCO DO COMERCIO EXTERIOR       9.3
                                   ---------------
PRODUTO NACIONAL LIQUIDO           896.3
DEPRECIACAO                        53.3
                                   ---------------
PRODUTO NACIONAL BRUTO             949.6
========================================
READY
```

2.6 EXERCÍCIOS DE APLICAÇÃO

1. Escrever os resultados produzidos pelos seguintes programas — Basic: (fornecer valores quaisquer pelo **INPUT**)

1.a)
```
100 PRINT "PROGRAMA-EXERCICIO"
200 PRINT "DATA: 31/JAN/1980"
300 PRINT "NOME: JOSE AGOSTINHO"
400 END
RUN
```

1.b)
```
10 PRINT "TITULO:..."
20 INPUT X, A
30 LET Y = X + A
40 PRINT X, A, Y
60 END
RUN
```

1.c)
```
10 INPUT N
20 PRINT "VALOR DE N = ", N
40 PRINT "VALOR DE N, N+N, N+N+N"
50 LET Y = N+N
60 LET Y1 = Y+N
70 PRINT N, Y, Y1
80 END
```

2. Escrever programas em Basic que calculem:

   2.a) a área do quadrado LxL para um valor de L lido.

   2.b) a área do círculo πR^2 e comprimento da circunferência $2 \pi R$ para valor de R lido.

   2.c) a expressão $(x^2 + y^2)/3$ para valores lidos de x e y.

3. Escrever um programa em Basic que escreva o nome e endereço seguintes:

   JOSE AUGUSTO XAVIER

   RUA JOAO X, 31

   S. PAULO – SP – CEP 00XX

4. Escrever um programa em Basic que leia e liste os valores do gasto mensal em:

VERDURAS	Cr$ 3 MIL
PADARIA	Cr$ 10 MIL
GASOLINA	Cr$ 55 MIL
LIVROS/REVISTAS	Cr$ 10 MIL
CARNES/AVES	Cr$ 15 MIL
ROUPAS	Cr$ 30 MIL
DIVERSOS	Cr$ 7 MIL

Fornecer, em conjunto com a listagem do nome dos itens e seus respectivos valores, os seguintes totais:

SUBTOTAL DE ALIMENTOS = Cr$

SUBTOTAL DE GASTOS ESSENCIAIS = ALIMENTOS + GASOLINA

SUBTOTAL DE GASTOS NÃO ESSENCIAIS = LIVROS/REVISTAS + ROUPAS + DIVERSOS

TOTAL GERAL DO GASTO DO MÊS = Cr$?

* *Sugestão:* Seguir, como modelo, o exercício do item 2.5.

Comandos de Repetição: FOR-NEXT 3

3.1 COMANDO DE REPETIÇÃO: FOR . . . NEXT

São usados em conjunto e permitem a repetição do trecho de programa compreendido entre o **FOR** e o **NEXT**. Equivalem ao comando **DO** do **FORTRAN**.

Exemplo:

```
20 FOR J = 1 TO 10 STEP 2
   . . . trecho do programa
30 NEXT J
```

Nesse exemplo, os comandos que formam o "trecho de programa" são executados 5 vezes, ou seja, para J = 1, 3, 5, 7 e 9; até atingir ou ultrapassar o valor limite 10. O incremento foi de 2 e, ao terminar a repetição, o programa prossegue após o comando **NEXT J**. Incremento igual a 1 pode ser subentendido.

Os valores iniciais, finais, e o incremento do contador de repetição J podem ser positivos, negativos, inteiros ou decimais ou, além disso, serem fornecidos através de expressões aritméticas.

Exemplo:

```
10 FOR K = 1 TO 10
20 FOR K = 1 TO X + Y STEP 0.53
50 FOR K = 10 TO 1 STEP -1
```

Observação: A variável K, que controla a repetição NÃO pode ser alterada por um comando **LET**.

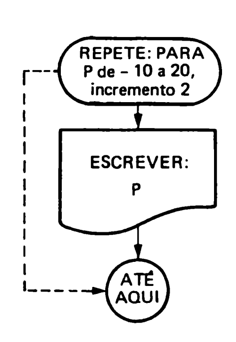
Exemplo:

```
50 FOR K = 1 TO 5
51 LET K = K + 2 (erro)
. . .
52 NEXT K
```

3.2 EXEMPLOS DE REPETIÇÃO OU "LOOPINGS" COM FOR-NEXT

Escrever os valores entre –10 e 20.

```
NEW
10 FOR P= –10 TO 20 STEP 2
20 PRINT P
30 NEXT P
40 END
RUN
```

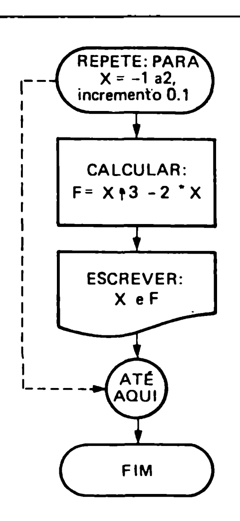


Resultado: Valores - 10, - 18, - 6,. . ., 18 e 20 escritos um em cada linha.

LISTAR os valores da função $f(x) = x^3 - 2x$ para os valores x variando de –1 a +2 com incremento de 0.1.

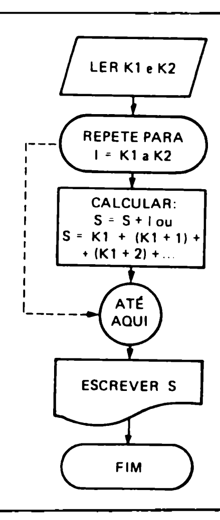
```
NEW
10 FOR X = –1 TO 2 STEP 0.1
20 LET F = X ↑ 3 – 2 * X
30 PRINT "'X = ",X," F(X) = ",F
40 NEXT X
50 END
RUN
```

```
X = - 1.000        F(X) = 1.000
X = - 0.900        F(X) = . . . . .
      (etc.)
X =   2.000        F(X) = 4.000
```



Escrever a soma de valores interiores entre K_1 e K_2

```
NEW
08 INPUT K1, K2
10 FOR I = K1 TO K2
20 LET S = S + I
30 NEXT I
40 PRINT S
50 END
```



Calcular a soma da série de Progressão Geométrica

$$S = 1 + a + a^2 + a^3 + a^4 + ... + a^N$$

e comparar o resultado com o valor da fórmula

$$S_1 = \frac{1 - a^{N+1}}{1 - a}$$

Observação: O termo a^k pode ser calculado como ((termo anterior a^{k+1}) vezes a), o que reduz o processo de cálculo.

```
NEW
10 LET S = 1
20 INPUT A, N
30 LET A1 = 1
40 FOR I = 1 TO N
50 LET A1 = A1 * A
60 LET S = S + A1
70 NEXT I
80 LET S1 = (1 – A ↑ (N + 1) )/(1 – A)
90 PRINT "SOMA DA SÉRIE IGUAL A", S
100 PRINT "SOMA PELA FÓRMULA IGUAL A", S1
110 END
```

Verificação manual com comentários:

```
 10 S = 1
 20 LER A, N               (por exemplo A = 2 e N = 3)
 30 A1 = 1
 40 PARA: I = 1           I=2            I=3
 50 A1 = 1 x 2 = 2        A1 = 2 x 2     A1 = 2 x 2 x 2
 60 S = 1 + 2             S = 1 + 2 + 2²   S = 1 + 2 + 2² + 2³ = 15
 70 Volta para comando 40 ou termina repetição
 80 S1 = (1 - 2⁴)/(1 - 2) = 15
 90 Escreve: SOMA DA SERIE IGUAL A 15
100 Escreve: SOMA PELA FORMULA IGUAL A 15
READY
```

> "Programa para calcular o fatorial do valor de 1 a 5."

	Comentários
NEW	
READY	
100 FOR K = 1 TO 5	• Início do ciclo de repetição para K = 1 a 5.
101 LET F1 = 1	
102 FOR I = 1 TO K	• Repetição para cálculo do fatorial de K.
103 LET F1 = F1*I	
104 NEXT I	• Fim do cálculo de um fatorial.
120 PRINT "FATORIAL DE", K, "EH IGUAL A", F1	• Escreve resultado.
130 NEXT K	• Volta para calcular mais um fatorial.
300 END	

Resultado:

```
RUN
FATORIAL DE   1   EH IGUAL A     1
FATORIAL DE   2   EH IGUAL A     2
FATORIAL DE   3   EH IGUAL A     6
FATORIAL DE   4   EH IGUAL A    24
FATORIAL DE   5   EH IGUAL A   120
READY
```

APLICAÇÃO COMERCIAL: PRESTAÇÕES E JUROS

Cálculo do valor da prestação variando a taxa de juros: A matemática financeira fornece a seguinte fórmula para calcular o valor da prestação A de uma compra a prazo de valor P, a ser paga em n vezes:

$$A = P \left[\frac{i\,(i+1)^n}{(1+i)^n - 1} \right]$$

onde P: valor presente da compra
i: taxa de juros

Por exemplo, o valor de cada prestação anual A para pagar uma compra de P = \$ 100.000,00, após 8 anos ao juro de 10% ao ano, é:

$$A = 100.000 \times \frac{0.1 \times 1.1^8}{(1.10)^8 - 1} = \$\ 18.744$$

Podemos calcular o valor da prestação A para qualquer taxa de juro i, variando de 6% até 12% ao ano.

Programa:

```
100 PRINT "VALOR DA PRESTACAO"
110 LET P=100000
120 LET N=8
130 PRINT "PARA PAGAR COMPRA DE $",P ; "CRUZEIROS"
140 PRINT "EM" ;N; "VEZES"
145 PRINT
150 FOR I=6 TO 12 STEP 2
152 LET K=I/100
155 LET J=(1+K)↑N
160 LET A=P*((K*J)/(J-1))
170 PRINT "JUROS DE" ;I; " % AO ANO PRESTAÇÃO ANUAL $", A
180 NEXT I
190 END
```

Resultados:

```
#RUN
VALOR DA PRESTACAO
PARA PAGAR COMPRA DE$  100000  CRUZEIROS
EM 8 VEZES
JUROS DE   6%AO ANO   PRESTAÇÃO ANUAL $   16103.5946
JUROS DE   8%AO ANO   PRESTAÇÃO ANUAL $   17401.4763
JUROS DE  10%AO ANO   PRESTAÇÃO ANUAL $   18744.4017
JUROS DE  12%AO ANO   PRESTAÇÃO ANUAL $   20130.2842

READY
```

3.3 REPETIÇÃO DENTRO DE OUTRA REPETIÇÃO (NESTED LOOP)

É freqüente a ocorrência de operação de repetição dentro de uma outra repetição. (Figura 3.1)

Neste caso é necessário observar algumas regras:

- Uma repetição **FOR-NEXT** deve sempre terminar antes da repetição **FOR - NEXT** que a contém.
- Dentro de uma repetição **FOR-NEXT**, é possível ocorrer várias outras repetições de mesmo nível, isto é, repetições que começam após o término completo da outra.

Exemplos:

Podemos ter os seguintes casos:

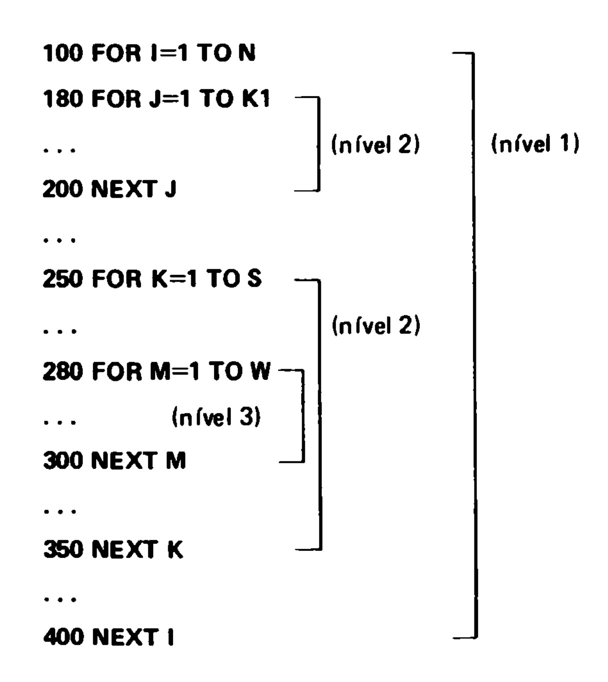
a) **FOR-NEXT** dentro de outro **FOR-NEXT**

```
100 FOR I=1 TO N
...
150 FOR J=1 TO K1
...
210 FOR K=2 TO S
...                    (etc. até um nível máximo permitido)
270 NEXT K
...
300 NEXT J
...
400 NEXT I
```

Exercício: Verificar o que faz o seguinte programa:

```
100 FOR I = 1 TO 3
200 FOR J = 1 TO 4
300 PRINT I, J, I + J
400 NEXT J
500 NEXT I
600 END
```

b) Vários **FOR-NEXT** de mesmo nível dentro de outro

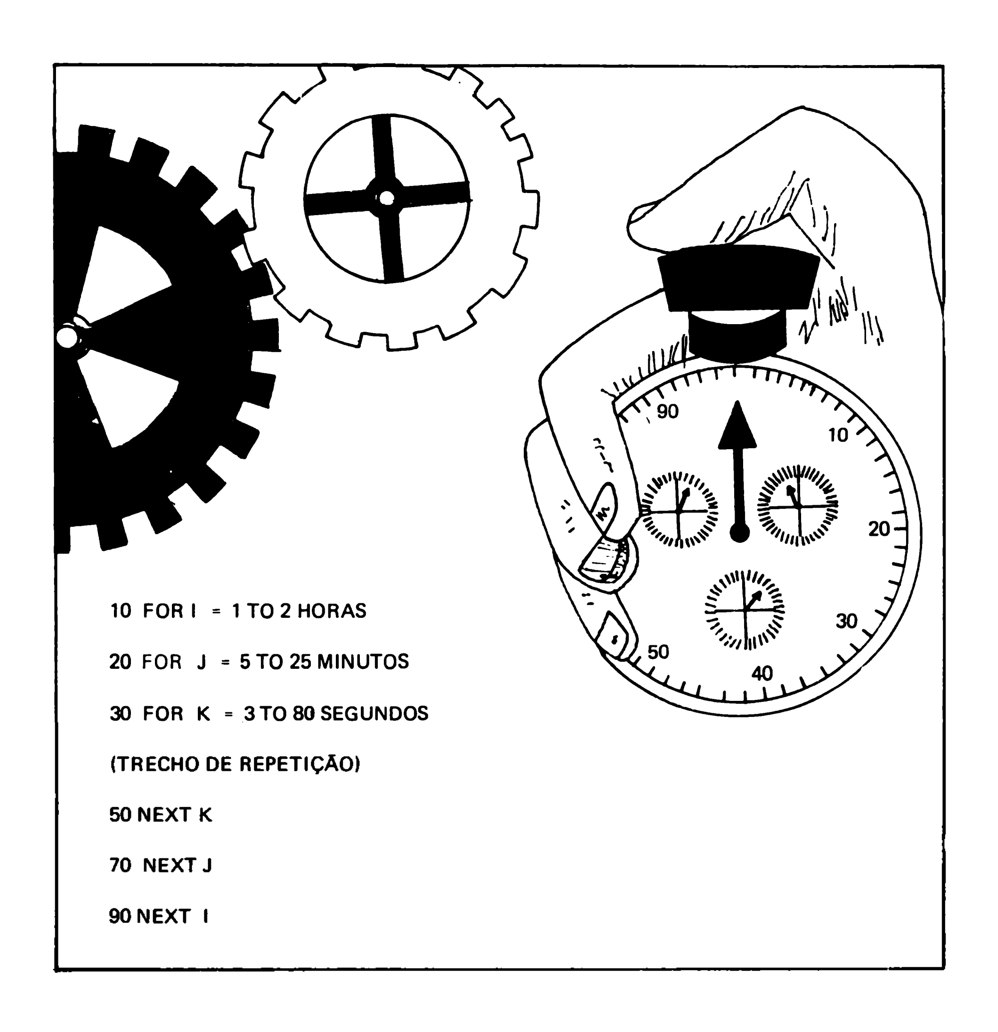


c) Repetições com erros

```
. . .
170 NEXT J
. . .
200 FOR K=1 TO K1
. . .
300 FOR M=2 TO S
. . .
400 NEXT K
. . .
500 NEXT M
```

(faltou **FOR J=. . .**)

(**FOR M=** não está totalmente contido em **FOR K=. . .**)



APLICAÇÃO COMERCIAL: PRESTAÇÕES E JUROS VARIADOS

Cálculo do valor da prestação variando juros e o número de prestações.

Na aplicação anterior podemos fazer variar o número N de prestações, de 6,8 até 10 vezes, e obter os valores das prestações para as diversas taxas de juros.

Acrescentamos o comando de variação de N:

FOR N=6 TO 10 STEP 2

Programa:

```
100 LET P= 100000
120 FOR N=6 TO 10 STEP 2
130 PRINT "N=";N;"VEZES"
145 PRINT
150 FOR I=6 TO 12 STEP 2
      (demais comandos são os mesmos da aplicação anterior)
190 END
```

Resultado:

```
NEW
N=6 VEZES

JUROS DE  6%AO ANO PRESTACAO ANUAL $...
JUROS DE  8%AO ANO PRESTACAO ANUAL $...
JUROS DE 10%AO ANO PRESTAÇAO ANUAL $...
JUROS DE 12%AO ANO PRESTACAO ANUAL $...

N = 8 VEZES

JUROS DE  6%AO ANO PRESTACAO ANUAL $...
   (etc.)

N = 10 VEZES
   (etc.)

READY
```

3.4 MATEMÁTICA FINANCEIRA: OUTRAS FÓRMULAS DE JUROS E PRESTAÇÕES

Sejam: i — taxa anual de juros

n — números de períodos

P — valor inicial atual ou valor presente do capital ou da compra.

F — valor futuro do capital após n períodos

A — valor de cada pagamento ou prestação paga em *n* vezes

A tabela de Fórmulas de Juros apresentada a seguir mostra as 6 fórmulas que fornecem a relação de equivalência entre os valores P, F e A, dado os valores *i* e *n*. (sem outras taxas como inflação e despesas).

O Programa Basic da aplicação comercial corresponde à fórmula 5.

Nº	Objetivo: Dado	Objetivo: Obter	Fórmula	Notação Abreviada	Explicação: Obter
1	P	F	$F=P(1+i)^n$	F=P(F/P,i,n)	Valor do capital F correspondente no fim de *n* períodos, ao valor P investido na data inicial.
2	F	P	$P=F\left[\frac{1}{(1+i)^n}\right]$	P=F(P/F,i,n)	Valor inicial P correspondente no fim de *n* períodos, ao valor F.
3	A	F	$F=A\left[\frac{(1+i)^n-1}{i}\right]$	F=A(F/A,i,n)	Valor F correspondente no fim de *n* períodos a uma série de *n* prestações A.
4	F	A	$A=F\left[\frac{1}{(1+i)^n-1}\right]$	A=F (A/F,i,n)	Valor de cada prestação A, a ser pago em *n* vezes, para capitalizar o montante F no fim do período *n*.
5	P	A	$A=P\left[\frac{i(1+i)^n}{(1+i)^n-1}\right]$	A=P(A/P,i,n)	Valor de cada prestação A, a ser paga *n* vezes, para amortizar o montante P devido na data inicial.
6	A	P	$P=A\left[\frac{(1+i)^n-1}{i(1+i)^n}\right]$	P=A(P/A,i,n)	Valor presente P, correspondente na data inicial, a uma série de *n* prestações A.

Nas notações abreviadas temos as seguintes relações de equivalência:

$$(F/P,i,n) = 1/(P/F,i,n) = (1+i)^n$$

$$(F/A,i,n) = 1/(A/F,i,n) = \left[\frac{(1+i)^n-1}{i}\right]$$

$$(A/P,i,n) = 1/(P/A,i,n) = \left[\frac{i(1+i)^n}{(1+i)^n-1}\right]$$

3.4.1 Uso das fórmulas: Exemplos

Fórmula 1:

Foi efetuado o empréstimo de Cr$ 200.000, à taxa de juros de 9% ao ano, e o pagamento deve ser feito em uma única parcela após 5 anos.

O valor a ser pago é:

$$F = P(1+1)^n = \text{Cr\$ } 200.000 \times (1+0{,}09)^5 = \text{Cr\$ } 200.000\ (F/P,9,5) =$$
$$= \text{Cr\$ } 200.000 \times (1{,}5386) = \text{Cr\$ } 307.724{,}00$$

Fórmula 2:

Queremos acumular Cr$ 800.000 em 6 anos, recebendo juros de 8 % ao ano.

O valor que devemos depositar hoje é:

$$P = F\,[1/(1+0{,}08)^6] = \text{Cr\$ } 800.000\ (P/F,8,6) = \text{Cr\$ } 800.000 \times (0{,}6302) =$$
$$= \text{Cr\$ } 504.160{,}00$$

Fórmula 3:

O pagamento da prestação de Cr$ 10.000 pago no fim de cada ano, durante 5 anos ao juro de 8 %, fornecerá o valor acumulado de:

$$F = \text{Cr\$ } 10.000\left[\frac{(1+0{,}08)^5 - 1}{0{,}08}\right] = \text{Cr\$ } 10.000 \times (F/A,8,5) = \text{Cr\$ } 10.000 \times$$
$$\times (5{,}8666) = \text{Cr\$ } 58.666{,}00$$

Fórmula 4:

O valor da prestação ou depósito anual necessário para produzir o valor acumulado de Cr$ 20.000 após 6 anos ao juro de 10 % é:

$$A = \text{Cr\$ } 20.000\left[\frac{0{,}10}{(1{,}10)^6 - 1}\right] = \text{Cr\$ } 20.000\ (A/F,10,6) = \text{Cr\$ } 20.000\,(0{,}1296) =$$
$$= \text{Cr\$ } 2.592{,}00$$

Fórmula 5:

Uma empresa investiu Cr$ 100.000 a taxa de 10 % ao ano, durante 8 anos. Esse capital pode ser recuperado mediante pagamentos anuais de:

$$A = \text{Cr\$ } 100.000 \left[\frac{0{,}1(1+0{,}1)^8}{(1+0{,}1)^8 - 1} \right] = \text{Cr\$ } 100.000 \text{ (A/P 10,8)} = \text{Cr\$ } 100.000 \text{ (0,1874)} = \text{Cr\$ } 18.740{,}00$$

Fórmula 6:

Qual o valor inicial necessário que permite, ao juro de 10 % ao ano, a retirada anual de Cr$ 1000, durante 10 anos?

$$P = \text{Cr\$ } 1.000 \left[\frac{(1+0{,}1)^{10} - 1}{(1+0{,}1)^{10} \times 0{,}1} \right] = \text{Cr\$ } 1.000 \text{ (P/A,10,10)} = \text{Cr\$ } 1.000 \text{ (6,1446)} = \text{Cr\$ } 6.144{,}60.$$

3.5 OBTENÇÃO DA TABELA DE JUROS E PRESTAÇÕES POR COMPUTADOR

Exercício:

Um programa simples de computador, permite o cálculo da tabela dos valores:

(F/P,i,n), (P/F,i,n), (F/A,i,n), (A/F,i,n), (A/P,i,n), (P/A,i,n)

para qualquer taxa de juros *i* e período *n*, fornecendo, por exemplo os seguintes resultados: (taxa de 8 % ao ano.)

Período n=1

1.0800	0.9259	1.0000	1.0000	1.0800	0.0259

Período n=2

1.1664	0.8573	2.0800	0.4808	0.5608	1.7833

Período n=3

1.2597	0.7938	3.2464	0.3080	0.3880	2.5771

(etc.)

Sugestão: Ver aplicação dada.

3.6 EXERCÍCIOS DE APLICAÇÃO

1. Escrever os resultados produzidos por cada um dos seguintes programas em Basic:

1.a)
```
10 FOR N = 1 TO 10
20 LET X = X + 1
30 LET Y = Y + N
40 NEXT N
50 PRINT X, Y
60 END
RUN
```

1.b)
```
10 FOR N = 1 TO 5
20 LET X = X + X * N
30 NEXT N
40 PRINT X
50 END
RUN
```

1.c)
```
10 FOR I = -5 TO 5 STEP 2.5
20 PRINT I
30 NEXT I
40 END
RUN
```

1.d)
```
10 FOR X = 1 TO 10 STEP 2
20 PRINT X
30 NEXT X
40 PRINT "BASIC"
50 END
RUN
```

1.e)
```
10 FOR X = 1 TO 20
20 FOR Y = 2 TO 10
30 FOR K = 1 TO 3
35 PRINT X, Y, K
40 NEXT K
50 NEXT Y
60 NEXT X
70 END
RUN
```

2. Escrever programas em Basic que calculem as seguintes expressões:

   2.a) $Y = 1 + 2 + 3 + 4 + 5 + \dots N$ para N dado.

   2.b) $Z = (x-1) + (x-2) + \dots + (x-N)$ para X e N dados.

   2.c) $W = U + U(U-1) + U(U-1)(U-2) + \dots + U(U-1)(U-2) + \dots + (U-N)$ para U e N dados.

   2.d) $S = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots + \frac{x^N}{N!}$ para x e N dados. (N ímpar)

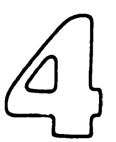
   Sugestão: O k-ésimo termo $\frac{x^k}{k!}$ = (termo anterior).$\left(\frac{x^2}{(k-1) \cdot k}\right)$

   por *Exemplo:* $\frac{x^5}{5!} = \frac{x^3}{3!} \cdot \frac{x^2}{4 \times 5}$

3. Escrever um programa em Basic que leia as quatro notas N1, N2, N3, e N4 de 20 alunos e forneça uma listagem de todas as notas e as médias de cada aluno.

4. Escrever o programa Basic solicitado na seção 3.5, para obtenção da tabela de juros e prestações.

Comandos de Desvio Condicional IF e Desvio Incondicional GO TO



4.1. COMANDOS IF. . .THEN

Usados na forma: **IF** (condição) **THEN** (comando).

Se a condição for verdadeira, executa o comando que está após o termo **THEN**; se for falsa, ignora esse comando e executa o comando seguinte. (Ver Figura 4.1.)

Exemplos:

```
105 IF X + Y = 56 + Z THEN PRINT "OBA-OBA"
130 IF A < 2 * B THEN GO TO 200
```

O comando GO TO 200 está, explicado na seção 4.5.

4.2 OPERAÇÕES RELACIONAIS

As condições possíveis são definidas pelos seguintes operadores relacionais:

=	igual	<=	menor ou igual
<	menor	>=	maior ou igual
>	maior	<>	diferente

As condições ou expressões relacionais sempre resultam em *verdadeiro* ou *falso*, e nunca fornecem resultados numéricos.

Exemplos:

X5 = 9	(verdadeiro ou falso?)
A1 <> (7+3*R)	(verdadeiro ou falso?)
A >= X + Y	(verdadeiro ou falso?)

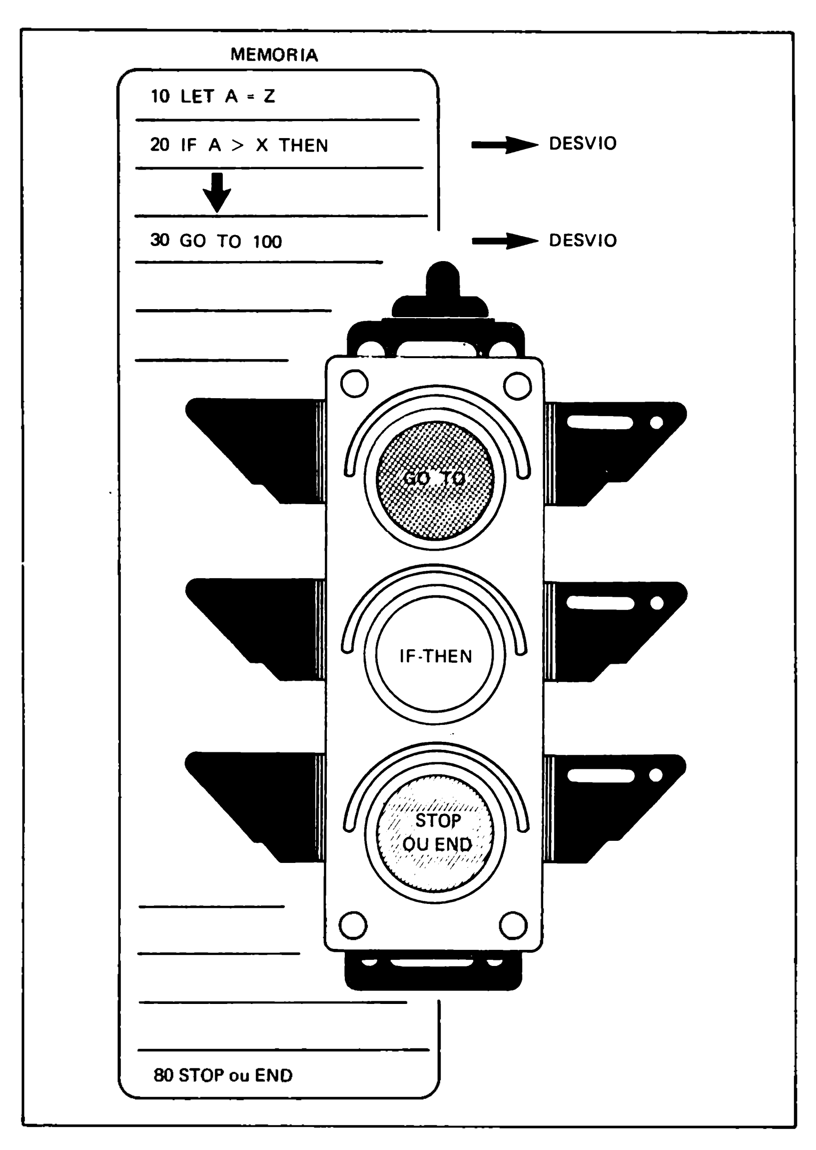


Figura 4.1 *Comandos IF-THEN e GO TO controlam desvios condicionais e incondicionais.*

4.3. OPERAÇÕES LÓGICAS AND, OR E NOT

Servem para ligar duas ou mais condições ou expressões relacionais ou modificar, através do **NOT**, o resultado de uma condição dada.

Formam sempre novas expressões relacionais cujo resultado é falso ou verdadeiro.

Exemplos:

(A< 3) AND (A> 0)	terá resultado verdadeiro se A< 3 e A > 0
(A< =B) OR (X> Y)	terá resultado verdadeiro se uma das condições (ou ambas) forem verdadeiras.
NOT (A = B)	equivale a A diferente de B.

```
30 IF A = B AND X = 0 THEN PRINT "VERDADEIRO"
```

4.4 COMANDO IF. . . THEN. . . ELSE

Algumas versões do Basic permitem o uso deste comando **IF**, que contém dentro do próprio comando a opção **THEN** a executar se a condição for verdadeira, e a opção **ELSE** a executar se a condição for falsa:

Exemplo:

```
05 INPUT X, Y
10 IF X = Y THEN PRINT "X IGUAL A Y" ELSE PRINT "X
DIFERENTE Y"
20 END
```

Resultado:

```
RUN
? 3, 5                    (valores de X e Y)
X DIFERENTE Y
READY
RUN
? 51 , 51
X IGUAL A Y
READY
```

Achar o valor máximo entre 10 valores lidos.

Comandos usados: **LET, FOR, NEXT, IF-THEN, GO TO.**

```
NEW
10 LET M = -1 E 10              (fixa valor negativo bem grande)
20 FOR I = 1 TO 10
30 INPUT X
40 IF M > X THEN GO TO 60       (Se M > X, o máximo continua sendo M)
50 LET M = X                    (Caso contrário, máximo é o novo valor X)
60 NEXT I
70 PRINT "MAXIMO M=",M
80 END
```

Resultados:

```
RUN
? 5                  (X=5 ,M=5)
? -30                (X=-30,M=5)
? 150                (X=150,M=150)
? 120                (X=120, M=150)
etc.
MAXIMO M=150         (ou qualquer valor maior lido)
READY
```

Achar o valor mínimo entre N valores lidos.

Comandos usados: **READ** e **DATA** no lugar de **INPUT**.

```
NEW
05 INPUT N
10 M1 = +1 E 10              (fixa valor positivo bem grande)
20 FOR I = 1 TO N
30 READ X
40 IF M1 < X THEN GO TO 60
50 LET M1 = X
60 NEXT I
70 PRINT "MINIMO IGUAL A", M1
80 DATA 35 , -10 , -5        (Escreve, separados por vírgulas, os N valo-
90 DATA 41 , 5 , 50          res a serem lidos pelo READ)
100 END
```

Resultados:

```
RUN
? 6                          (valor de N=6)
MINIMO IGUAL A -10
READY
```

Observação: Comandos **READ** e **DATA** serão vistos no capítulo seguinte.

4.5 COMANDO GO TO n

Esse comando desvia a execução do programa para o comando de número *n*. A execução do programa continua seqüencialmente a partir da linha *n*. O número *n* pode ser maior ou menor do que a linha em que está o comando.

Exemplo:

```
10 INPUT A,B,C
20 LET D = A + B + C
40 PRINT A,B,C
50 PRINT D
55 GO TO 10
60 END
```

Resultado:

```
RUN
? 1,2,3                    (valores A,B,C)
1        2        3
6                          (D)
?                          (aguarda novos valores de A, B, e C)
```

Observação: Com o uso do **GOTO** o exemplo visto sempre permance no estado de execução **RUN** aguardando novos valores A,B e C. Para sair deste estado é necessário usar a tecla apropriada (**RESET, BREAK** ou equivalente) e assim teremos a palavra **READY.**

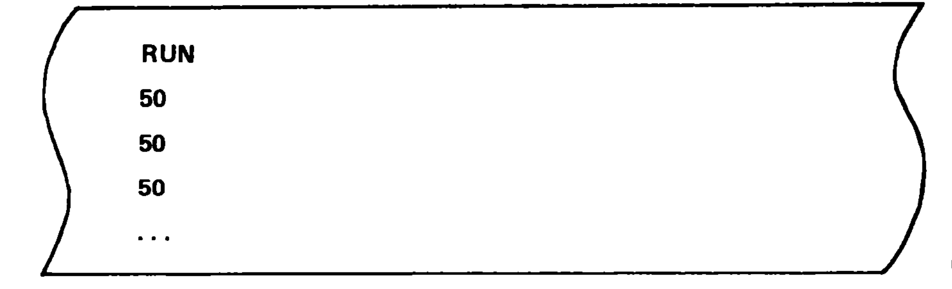
O uso desnecessário do comando **GOTO** deve ser evitado.

4.5.1 Uso da tecla BREAK para interromper a repetição (loop)

Exemplo:

```
10 A = 5
20 B = 10
30 C = A *B
40 PRINT C
50 GOTO 10
60 END
```

Resultado:



```
RUN
50
50
50
...
```

No programa anterior o comando **GOTO** executa um "loop" (repetição) infinito, e isto em geral não é desejado. A única maneira de se parar o programa anterior é através da tecla **BREAK**, e o programa será interrompido, indicando a mensagem:

BREAK IN N

onde N indica o nº da linha em que o programa foi interrompido.

Observação: Nem todos os micromputadores possuem a tecla **BREAK**. Pode-se usar o acionamento simultâneo de duas teclas: o **CRTL** (Control) e a tecla **'C'**.

4.5.2. Outras formas possíveis do **GOTO**

O comando **GOTO** pode ser digitado de três maneiras:

1) **GOTO** (sem espaço entre o **GO** e o **TO**)
2) **GO TO** (com espaço entre o **GO** e o **TO**)
3) Pode ser eliminado se usado dentro da opção **THEN** do comando **IF**.

Exemplos:

50 IF X = Y THEN GO TO 120

ou

50 IF X = Y THEN GOTO 120

ou

50 IF X = Y THEN 120

4.6. EXERCÍCIOS DE APLICAÇÃO

1. Para valor de A = 6, 3 e 5, escrever o resultado fornecido pelos programas.

1.a)
```
20 INPUT A
30 IF A = 5 THEN GO TO 70
40 PRINT A
50 GO TO 20
70 PRINT A, "TERMINOU"
80 END
RUN
```

1.b)
```
10 INPUT A
20 IF A > 5 THEN 60
30 IF A = 5 THEN 80
40 PRINT "A MENOR QUE CINCO", A
50 GO TO 90
60 PRINT "A MAIOR QUE CINCO",A
70 GO TO 90
80 PRINT "A  IGUAL A CINCO",A
90 END
RUN
```

2. Para valores de A = 5 B = - 3 e C = 1, dizer qual o resultado de cada uma das expressões lógicas:

2.a) (A > = 3) OR (B = 5) OR (C < 2)

2.b) (A * B < = 10) AND (A – C > 2)

2.c) NOT (A * B < = 10)

2.d) ((A * B < = 10) AND (B = C)) OR (A + B + C > 5)

3. Escrever um único programa que calcule o valor da seguinte função, para X lido:

F = X + 2	se X < 1
F = X * X + 1	se 1 ≤ X ≤ 5
F = X * (X + 1)	se X > 5

Testar manualmente o programa para valores X = 1, 4 e 7.

4. O conceito final da matéria de uma escola é dado pelo seguinte critério:

Nota de 90 a 100 :	Conceito A
Nota de 75 a 89,9 :	Conceito B
Nota de 60 a 74,9 :	Conceito C
Nota de 50 a 59,9 :	Conceito I
Nota inferior a 50 :	Conceito R

Escrever um programa em Basic que leia as notas de 5 alunos e escreva o conceito correspondente.

Testar manualmente o programa para as notas: 100, 89, 77, 91 e 45.

Comando de Leitura: READ, DATA e RESTORE

5

5.1 LEITURA DE DADOS PELO PAR DE COMANDOS READ E DATA READ.

5.1.1 READ

O comando **READ** faz com que os dados sejam lidos de um **buffer** definido pelo comando **DATA**. Os valores do **DATA** são atribuídos a cada variável do comando **READ**, sucessivamente.

Observação: buffer significa local intermediário de armazenamento de dados.

DATA

O comando **DATA** armazena os dados como parte do programa. Esse comando não é propriamente executado, especificando apenas o conjunto de dados necessário para os comandos **READ** que se encontram no programa.

Os comandos **DATA** podem ser colocados em qualquer parte do programa (geralmente são colocados no fim) e serão lidos seqüencialmente conforme necessário.

5.1.2 READ e DATA

Esses dois comandos devem atuar em conjunto e possuem certa semelhança com o comando **INPUT**; exceto que o conjunto de dados em vez de entrar através do teclado do terminal, é colocado dentro do próprio programa, ou seja, indicamos no comando **READ** as variáveis que devem receber os valores e, construímos uma tabela de valores numéricos ou *strings* de caracteres no comando **DATA**.

A atribuição de valores (dos comandos **DATA** às variáveis dos comandos **READ**) é efetuada de modo seqüencial, a partir dos primeiros dados e variáveis, da esquerda para a direita.

Não é necessário usar um comando **DATA** para cada comando **READ**, pois a atribuição de valores é efetuada de variável em variável, de dado em dado, controlando-se apenas a seqüência e a quantidade dos valores atribuídos. Um contador interno controla a seqüência de leitura.

Os dados de um único comando DATA podem servir a vários comandos **READ** e vice-versa.

Leitura simples

```
10 READ A, B, C
20 PRINT A, B, C
30 DATA 10, 20, 3.5
40 END
```

Resultado:

```
RUN
10        20        3.5          (valores das variáveis A, B e C)
READY
```

Leitura com falta de dado

```
10 READ A1, B1, C1
20 PRINT A1, B1, C1
30 DATA 10, 20
40 END
```

Resultado:

```
RUN
10        20
ERROR IN 10              (falta de dado  para a variável C1)
```

Leitura com excesso de dado

```
10 READ A1, B1, C1
20 PRINT A1, B1, C1
30 DATA 10, 20, 30, 40, 50
40 END
```

Resultado:

```
RUN
10      20      30
READY                 (os dados 40 e 50 não foram lidos)
```

Vários **READ** *e um só* **DATA**

```
10 READ A1, B1, C1
20 PRINT A1, B1, C1
30 READ D, E
35 PRINT D, E
40 DATA 10, 20, 30, 40, 50
50 END
```

Resultado:

```
RUN
10      20      30
40      50
READY                 (os dados lidos por dois READ estão em
                      um só DATA)
```

Leitura de strings

```
10 READ A$, A1$, A2$
20 PRINT A$, A1$, A2$
30 DATA"BASIC", "15", "COMPUTADOR"
40 END
```

Resultado:

```
RUN
BASIC          15          COMPUTADOR
READY
```

5.2 COMANDO **RESTORE**

Quando o comando **READ** termina de ler todos os dados contidos no comando **DATA**, esses dados não podem ser relidos.

O comando **RESTORE** foi criado justamente para eliminar esta limitação, se dentro de um programa desejamos repetir a leitura dos mesmos dados contidos nos **DATA**. O comando **RESTORE** zera o contador interno de dados, que aponta novamente o primeiro valor do primeiro comando **DATA**, podendo todo o conjunto de dados ser relido desde o início, através de novos comandos **READ**. Não é possível voltar para um dado intermediário do **DATA**.

```
10 READ A,B          • (Ler valores 1 e 2)
20 PRINT A,B
30 RESTORE           • (Reinicia contador de dados)
40 READ C, D         • (Ler valores 1 e 2)
50 PRINT C,D
60 DATA 1,2          • (Dados lidos duas vezes)
70 END
```

Resultado:

```
RUN
1       2
1       2
READY
```

Existem versões de Basic onde o comando **RESTORE** pode ser usado da seguinte maneira:

RESTORE nn

onde 'nn' representa o número de linha que contém o comando **DATA,** a partir de onde desejamos reiniciar a leitura.

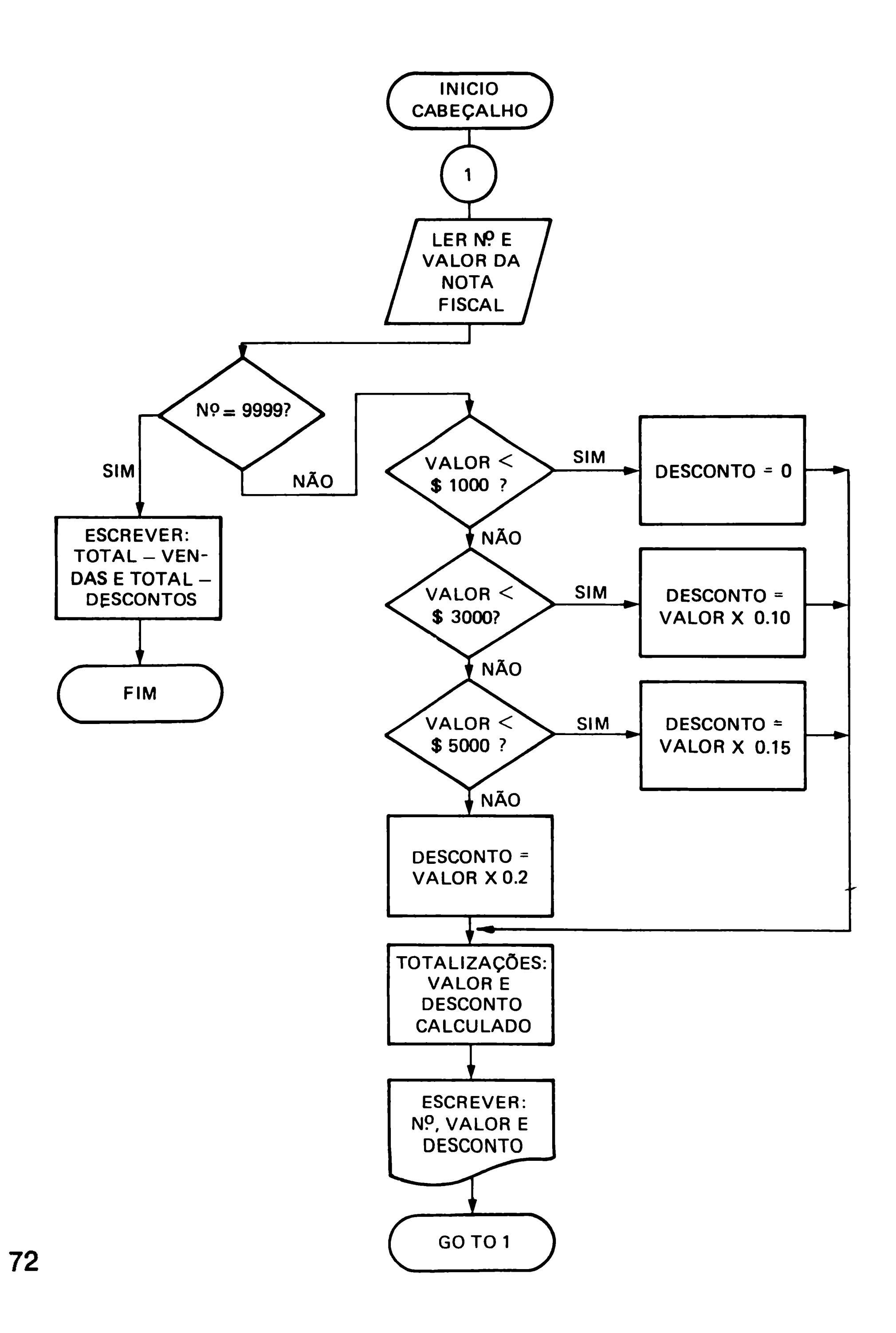
5.3 APLICAÇÃO COMERCIAL: DESCONTO EM NOTA FISCAL

Cálculo de Desconto em Nota Fiscal

Uma loja oferece os seguintes descontos sobre o valor da Nota Fiscal de compras efetuadas:

valor	desconto
até $ 999	0%
de $1000 a 2999	10%
de $3000 a 4999	15%
mais de $5000	20%

Escrever um programa Basic que liste o nº e valor das notas fiscais e os respectivos descontos calculados. Totalizar o valor das notas e os descontos. O nº 9999 de nota fiscal indica fim de operação.


INICIO
CABEÇALHO
1
LER Nº E
VALOR DA
NOTA
FISCAL
Nº = 9999?
SIM
NÃO
ESCREVER:
TOTAL – VEN-
DAS E TOTAL –
DESCONTOS
FIM
VALOR <
$ 1000 ?
SIM
DESCONTO = 0
NÃO
VALOR <
$ 3000?
SIM
DESCONTO =
VALOR X 0.10
NÃO
VALOR <
$ 5000 ?
SIM
DESCONTO =
VALOR X 0.15
NÃO
DESCONTO =
VALOR X 0.2
TOTALIZAÇÕES:
VALOR E
DESCONTO
CALCULADO
ESCREVER:
Nº, VALOR E
DESCONTO
GO TO 1

Programa

```
0100 PRINT "CALCULO DE DESCONTOS EM NOTA FISCAL"
0110 PRINT
0120 PRINT "NUMERO      VALOR DA NOTA          DESCONTO"
0130 PRINT
0140 READ N,V
0150 IF N = 9999 THEN 340
0160 IF V< 1000 THEN 220
0170 IF V< 3000 THEN 240
0180 IF V< 5000 THEN 260
0190 REM DESCONTO DE 20 %
0200 LET D = V *0.2
0209 REM DESCONTO ZERO
0210 GO TO 280
0220 LET D=0
0230 GOTO 280
0239 REM DESCONTO DE 10 %
0240 LET D = V * 0.1
0250 GOTO 280
0259 REM DESCONTO DE 15%
0260 LET D = V * 0.15
0270 REM TOTALIZACOES
0280 LET T1 = T1 + V
0290 LET T2 = T2 + D
0300 PRINT N, V, D
0310 GOTO 140
0320 REM FIM DE CALCULO
0330 REM ESCREVE TOTAIS
0340 PRINT "===================================="
0350 PRINT "TOTAIS $", T1, T2
```

```
0360 DATA 357, 800
0370 DATA 410, 1200
0380 DATA 1347, 1500
0390 DATA 1511, 6000
0400 DATA 9999, 9999
0600 END
READY
#
```

Resultados:

```
#RUN
CALCULO DE DESCONTOS EM NOTA FISCAL
NUMERO     VALOR DA NOTA     DESCONTO
357             800              0
410             1200             120
1347            1500             150
1511            6000             1200
======================================
TOTAIS $        9500             1470
READY

#
```

5.4 EXERCÍCIOS DE APLICAÇÃO

1. Explicar, com exemplo, a função dos comandos **READ, DATA** e **RESTORE**. Qual a diferença entre os comandos **READ** e **INPUT**?

2. Dizer quais os resultados dos seguintes programas:

2.a)
```
 50 DATA 33, 55, -1, 43, 51
 60 READ Z, X
 70 READ Y, W
 80 PRINT Y, W, Z, X
 90 READ X
100 PRINT Y, W, Z, X
110 END
    RUN
```

2.b)
```
10 DATA 10
20 READ X,Y
30 DATA 5
40 DATA 50, 100
50 READ Z
70 PRINT X, Y, Z
80 END
   RUN
```

2.c)
```
 10 READ X, Y, Z
 20 RESTORE
 30 PRINT X, Y, Z
 35 READ W
 40 READ X, Y, Z
 50 PRINT X, Y, Z
 60 RESTORE
 70 READ X, Y, Z
 80 PRINT X, Y, W, Z
 90 DATA 5, 3, 4, 9
100 DATA 7, 7, 8, 15, 21
110 END
```

3. Temos os seguintes valores referentes a uma classe de alunos: nº e notas:

Nº do aluno	Física	Matemática	Português	Química
347	10	8	10	7
348	9	8	10	7
450	7	8	8	7
510	8	9	10	7
610	10	6	9	6
734	5	4	7	7
859	4	7	8	8

Escrever um programa que leia esses valores (através do comando **READ** e **DATA**) e forneça uma listagem contendo: nº, notas e média de cada aluno. Após a listagem das notas escrever as médias da classe em cada uma das matérias.

Comando PRINT: Uso do (?), (;) e <PRINT 6

6.1 UTILIZAÇÃO DE *VÍRGULA* E *PONTO-E-VÍRGULA* COM O COMANDO **PRINT**

O comando **PRINT** faz uso de vírgula (,) e ponto-e-vírgula (;) para fazer o controle de impressão de mensagens na linha da tela ou impressora.

Suponha o interpretador Basic que possui uma linha de impressão de 72 caracteres, ou seja, cada linha da tela ou da impressora pode conter 72 caracteres. Esta linha de impressão normalmente é dividida em quatro zonas de 18 caracteres cada uma que, somadas, completam os 72 caracteres. Veja o esboço a seguir:

Coluna

1	18-19	36-37	54-55
ZONA 1	ZONA 2	ZONA 3	ZONA 4

O uso de uma vírgula separando as mensagens ou variáveis a serem impressas faz com que o Basic inicie a impressão na zona seguinte disponível. O ponto-e-vírgula coloca o valor a ser impresso na posição (e não na zona) imediata após a última impressão.

PRINT	com vírgula e ponto-e-vírgula

a)
```
10 PRINT "COMPILADOR", "INTERPRETADOR"
20 END
```

```
RUN
COMPILADOR              INTERPRETADOR
READY
(Coluna 1)              (col. de 19. . . 31)
```

b) 10 PRINT "COMPILADOR; "INTERPRETADOR"

20 END

```
RUN
COMPILADORINTERPRETADOR   (escreve imediatamente após
                          primeiro dado)
READY
```

c) 10 PRINT "COMPILADOR";

20 PRINT "INTERPRETADOR"

30 END

```
RUN

COMPILADORINTERPRETADOR
READY
```

No exemplo (c), apesar dos comentários terem sido colocados em linhas de comandos diferentes, o ponto-e-vírgula colocado após a mensagem **"COMPILADOR"** fez com que o Basic imprimisse a segunda mensagem na mesma linha suspendendo a impressão no primeiro comando. Se na linha 10 não fosse colocado o ponto-e-vírgula após a mensagem **"COMPILADOR"**, teríamos duas operações de impressão.

A utilização do ponto-e-vírgula (;) separando os comentários ou valores faz com que o Basic não busque a próxima zona de impressão ou não execute a impressão, colocando-os um após a outro.

d) Impressão de valores seguidos.

Por exemplo, supondo que X = 15 e Y = 25, teremos:

PRINT X;Y	• imprimirá **1525** sem espaço entre os valores;
PRINT X,Y;	• nada imprimirá, aguardando o próximo comando PRINTX
PRINT X	• teremos a impressão de **15** (X) **25** (Y) **15** (X)

e) Impressão de pergunta e resposta.

```
10 PRINT "QUAL E' O SEU NOME?";
20 INPUT A$
30 PRINT "BOM DIA"; A$
40 END
```

```
RUN
QUAL E' O SEU NOME? ? MARIO     ("tecla enter ou return")
BOM DIA MARIO
READY
```

6.2 TRAÇADOR DE GRÁFICOS

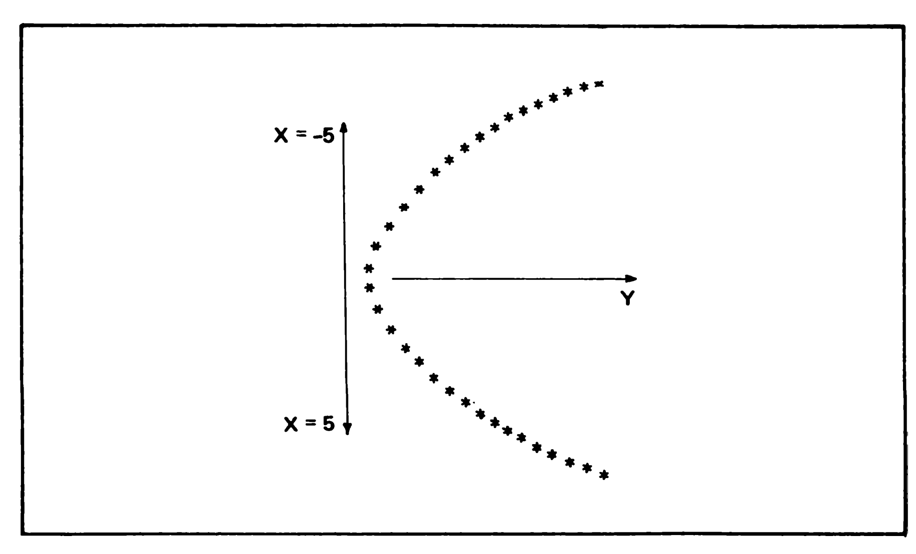
Traçamento do gráfico da função Y = X ** 2 + 1.

```
NEW
READY
100 FOR X = – 5 TO 5 STEP .5
102 LET Y = X * X + 1
104 FOR K = 1 TOY
105 PRINT "ƀ";
106 NEXT K
107 PRINT " *"
108 NEXT X
109 END
```

Linhas 104 a 107: Imprime Y espaços em branco numa mesma linha e então o sinal " * "

(ƀ) – representa espaço em branco

RUN



6.3 USO DO SINAL DE INTERROGAÇÃO (?) NO LUGAR DA PALAVRA PRINT

Existem versões Basic onde o comando **PRINT** pode ser substituído por uma interrogação (?). Em vez de se escrever a palavra **PRINT**, basta escrever o símbolo de interrogação (?).

PRINT COM "?"

```
10 ? "MEMORIA"
20 ? "UNIDADE CENTRAL DE PROCESSAMENTO"
30 END
```

```
RUN
MEMORIA
UNIDADE CENTRAL DE PROCESSAMENTO
READY
```

Observação: Durante uma listagem do programa pelo comando **LIST**, será sempre impressa a palavra **PRINT**, mesmo tendo sido usado o sinal (?).

LISTAGEM DE "?"

```
10 ? "MEMORIA"
20 ? "UNIDADE CENTRAL DE PROCESSAMENTO"
30 END
```

```
LIST
10 PRINT "MEMORIA"
20 PRINT "UNIDADE CENTRAL DE PROCESSAMENTO"
30 END
READY
```

6.4 A FUNÇÃO ESPECIAL **CHR$ (N)**

Podemos imprimir qualquer caractere do código ASCII utilizando a função especial **CHR$ (N)**.

O valor de N varia de 0 até 255 e a cada valor numérico compreendido nessa faixa corresponde um caractere ASCII (ASCII – American Standard Code for Information Interchange).

Listagem dos caracteres ASCII: código 32 em diante.

```
10 FOR I = 32 TO 255
40 PRINT I, CHR$(I)
80 NEXT I
90 END
```

```
RUN
32      "b̸"     (espaço)
33       !
34       "
35      #
36       $
37      %
38      &
39       '
40      (
41      )
42       *
43      +
etc.
```

Observação: O programa deve ser melhorado para escrever os caracteres em mais colunas e assim economizar tempo e reduzir o uso de papel se for usada uma impressora. (Ver tabela ASCII no Apêndice B).

Impressão de comentários com aspas.

Quando queremos forçar a impressão de aspas (que são usadas para separar comentários e por isso não são impressos) usamos **CHR$(34)**.

```
10 PRINT CHR$ (34); "ORDEM E PROGRESSO"; CHR$ (34)
20 END
```

```
RUN
"ORDEM E PROGRESSO"
READY
```

Em algumas versões de Basic, para se colocar um *string* entre aspas é utilizado um outro recurso: o de duplas aspas (" ").

```
10 PRINT " " "ORDEM E PROGRESSO" " "
20 END
```

```
RUN
"ORDEM E PROGRESSO"
READY
```

6.5 MODO DIRETO DE IMPRESSÃO: USO DO **PRINT** COMO UMA CALCULADORA

O comando **PRINT** pode imprimir diretamente (sem esperar o comando **RUN** para executar o programa inteiro) valores ou resultados obtidos através de uma expressão aritmética. Basta usá-lo sem o número de comando.

Operação direta com PRINT

```
PRINT 10
10
READY
```

```
PRINT 10, 20, 30
10          20          30
READY
```

```
PRINT 5 + 2
7
READY
```

```
PRINT 5 + 2, 7 + 3, 8 *2
          7    10    16
```

impressão de resultado de expressões aritméticas.

6.6. COMANDO< PRINT PARA ESCREVER NO FORMULÁRIO DA IMPRESSORA (OU LPRINT)

Esse comando é utilizado quando se deseja imprimir valores numéricos e **strings** em uma impressora em vez da tela de um terminal de vídeo. Pode ser usado com todas as atribuições do comando **PRINT** propriamente dito, ou seja, todas as características do comando **PRINT** podem ser usadas no comando **< PRINT**.

Exemplo:

Se digitarmos um programa através do teclado de um terminal de vídeo e substituirmos o comando **PRINT** por **< PRINT**, todos os resultados serão apresentados em uma impressora.

```
10 A = 3 + 2 * 3
20< PRINT A
30 END
```

```
RUN
9          (valor da variável impresso em uma impressora)
READY      (o comando retorna automaticamente ao teclado do
           monitor de vídeo)
```

6.7 EXERCÍCIOS DE APLICAÇÃO

1. Escrever os resultados fornecidos pelos seguintes programas.

1.a)

```
10 A$ = "BASIC ƀ"
20 B$ = "PARA ƀ"
30 C1$ = "ENGENHEIROS"
40 PRINT A$; B$; C1$
50 END
RUN
```

1.b)

```
10 READ A$, B$, B1$
20 PRINT A$; B$, B1$
30 READ B$, B1$
40 PRINT A$, B$, B1$
50 DATA "BASIC ƀ","NIVEL II ƀ","1983", "BASIC", "1984"
60 END
RUN
```

1.c)
```
10 A$ = "O VALOR DE"
20 B$ = "D2"
30 C$ = "E IGUAL A"
40 A1 = 5
50 B1 = 3
60 C1 = 2
70 D2 = A1 + B1 + C1
80 PRINT A1, B1, C1
90 PRINT A$, B$, C$, D2
100 END
RUN
```

1.d)
```
10 A = 5
20 B = 3
30 C1 = 2
40 D2 = A + B + C1
50 PRINT "O VALOR DE D2 E IGUAL A"; D2
60 END
RUN
```

1.e)
```
10 PRINT "A = B * C – D"
20 PRINT "VALOR DE B IGUAL A";
30 INPUT B
40 PRINT "VALOR DE C IGUAL A";
50 INPUT C
60 PRINT "VALOR DE D IGUAL A";
70 INPUT D
80 A = B * C – D
90 PRINT A, B, C, D
100 END
RUN
```

2. Dizer por que ocorreu a mensagem de erro nos seguintes programas:

2.a)
```
10 READ A1, A2, A3, B1, B2, B3
20 PRINT A1, A2, A3, B1, B2, B3
30 READ B1, B2, B3
35 PRINT B1, B2, B3
40 DATA 1, 2, 3, 4, 5, 6
50 DATA 7, 8
60 END
```

```
RUN
 1          2          3          4        5        6
ERROR IN 30
READY
```

2.b)
```
10 READ A$, B, C$
20 PRINT A$, B, C$
30 READ B, C$
40 PRINT A$, B, C$
50 DATA "BASIC", 1983, "NIVEL II", BASIC II, 1983
```

```
RUN
BASIC        1983        NIVEL II
ERROR IN 30
   READY
```

2.c)
```
10 READ A$, B, C$
20 PRINT A$, B, C$,
30 READ B, C$
40 PRINT A$, B, C $
50 DATA "BASIC", 1983 "NIVEL II", 1983, "BASIC"
```

```
RUN
ERROR IN 20
READY
```

3. No exercício de — traçamento de gráfico — queremos efetuar as seguintes modificações:

   3.a) Escrever, na respectiva linha correspondente, os valores de X e Y.

   3.b) Alterar cada barra do gráfico para barra contínua do tipo:

   (=========== *)

4. Traçar o gráfico da função

   $y = x^3 + 1$ para x de 1 a 5.

   Reduzir a escala para 1/5.

   (Sugestão: adaptar o exercício da seção 6.2.)

Variáveis Dimensionadas: Matrizes e Vetores



7.1 VARIÁVEIS DIMENSIONADAS OU MATRIZES

Para designar um conjunto de valores de uma tabela ou matriz, são usadas as variáveis dimensionadas. São definidas por um nome formado por uma só letra e índices colocados entre parênteses. As matrizes podem ser de uma ou de duas dimensões e os índices podem ser constantes ou variáveis simples. As matrizes de uma dimensão são denominadas *vetores.* (Ver Figura 7.1.)

Exemplos:

A(1), W(100), X(1,2), B(M,3), Z(L,K)

Nota: Algumas versões do Basic permitem usar índice zero.

Exemplo:

A(0), A(1), A(2), A(3), . . ., B(0,1), etc.

Comando **DIM**

DIM	Reserva espaço na memória para as matrizes usadas no programa.

Exemplo:

110 DIM X(15), A(5,5)	(definiu um vetor X com 15 elementos e uma matriz A com 5 x 5 elementos)
100 DIM X(N)	(errado, pois N deve ter um valor numérico que informe o número total de elementos X(I).)

Nota importante: é sempre recomendável colocar os comandos **DIM** no início do programa. Em qualquer outra linguagem (como FORTRAN, PL/1, PASCAL) essa exigência é obrigatória. O Basic aceita **DIM** no meio do programa ou mesmo na forma (**10 INPUT N, 20 DIM X(N)**) que, entretanto, pode dar MENSAGEM DE ERRO na execução repetida desses comandos.

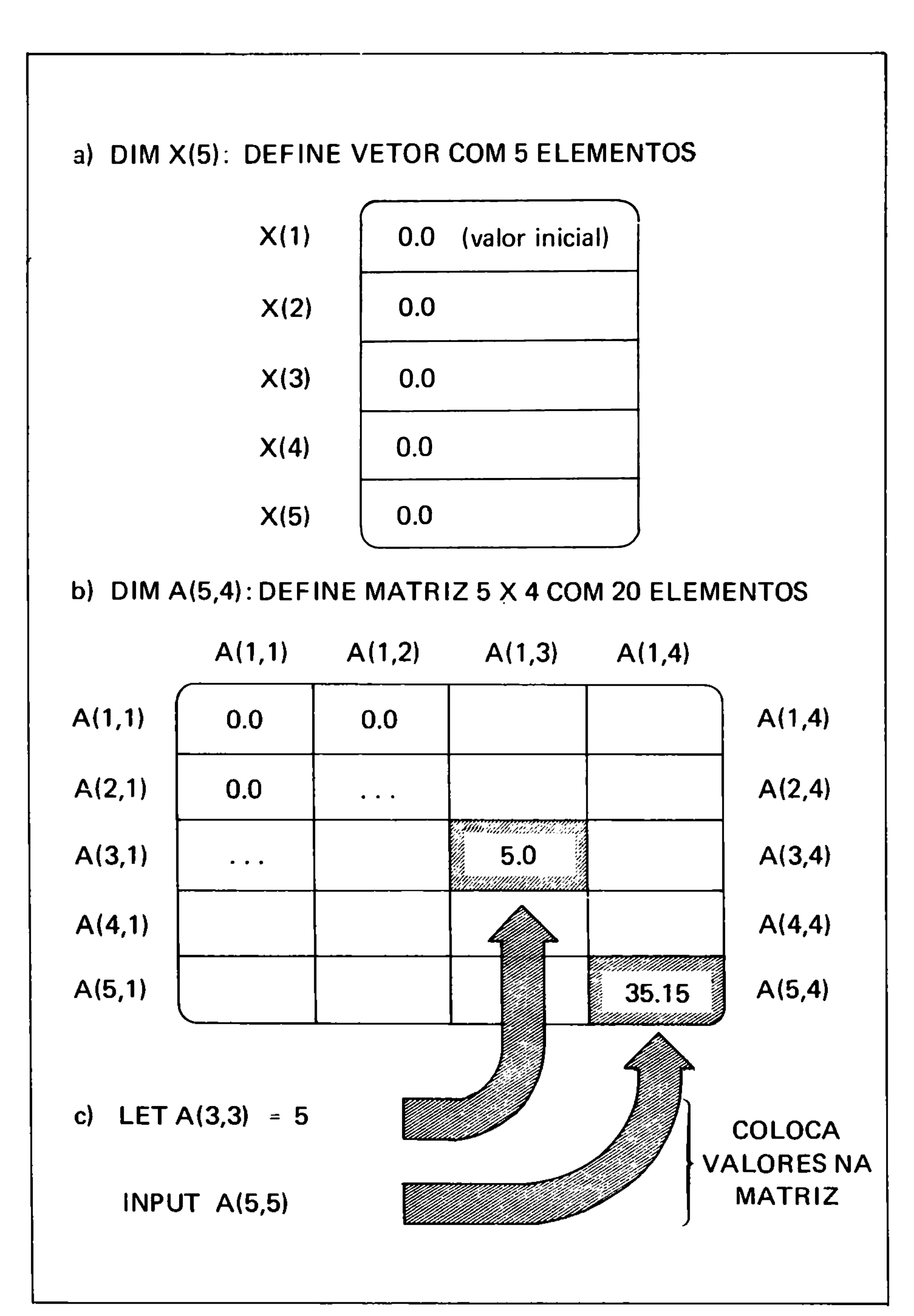
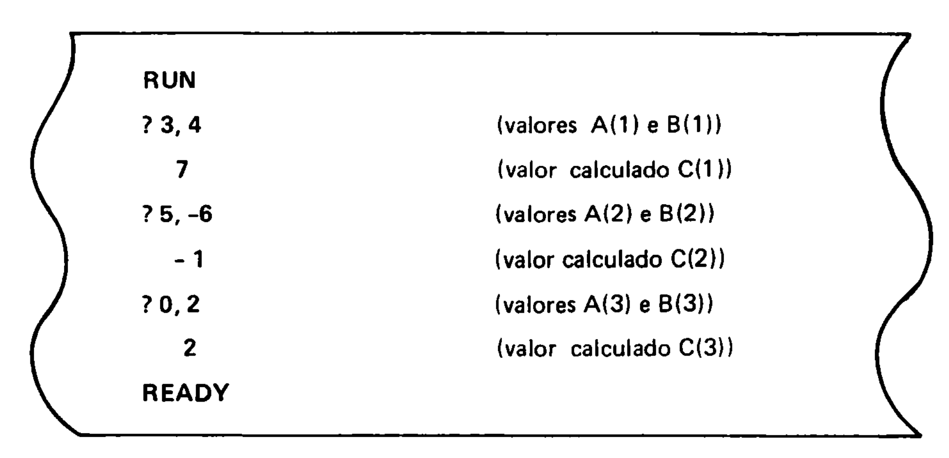


 Figura 7.1 – Variáveis Dimensionadas.

7.1.1 Programa para calcular a soma de dois vetores, A e B

NEW	Comentários
READY	
100 DIM A(3), B(3), C(3)	• Define local para vetores A,B e soma C
110 FOR I = 1 TO 3	• Início da repetição
120 INPUT A(I), B(I)	• Leitura de um elemento de A e de B
130 LET C(I) = A(I) +B(I)	• Calcula um elemento do vetor resultado
140 PRINT C(I)	• Escreve o resultado
150 NEXT I	• Volta para ler novos elementos
END	

Resultados:



RUN	
? 3, 4	(valores A(1) e B(1))
7	(valor calculado C(1))
? 5, -6	(valores A(2) e B(2))
- 1	(valor calculado C(2))
? 0, 2	(valores A(3) e B(3))
2	(valor calculado C(3))
READY	

Observação: Para vetor ou matriz, de dimensão *até dez*, o uso do comando **DIM** é facultativo, pois o Basic define automaticamente o vetor ou matriz de 10 ou 10 x 10 elementos, à medida que usamos uma variável dimensionada. Então o comando **100 DIM A(3), B(3), C(3)** é dispensável. Para vetor ou matriz com dimensão maior que 10 ou 10 x 10 é necessário usar o comando **DIM.**

Exemplo:

100 A(20), X(20, 15)

7.1.2 Cálculo da média aritmética $S = \frac{\Sigma V(I)}{N}$ (I = 1, 2, . . ., N)

Calcular a média aritmética $S = \frac{\Sigma V(I)}{N}$ (I = 1, 2, . . ., N)

e a variância ou quadrado da dispersão média $D = \frac{\Sigma (V(I) - S)^2}{N - 1}$

de N = 10 valores do vetor V = {V (1), V(2), . . ., V(10)}

```
NEW
05 DIM V(10)                (Define área da memória para o vetor)
30 FOR 1 = 1 TO 10
40 READ V(I)                (Cálculo da média aritmética)
50 LET S = S + V(I)
60 NEXT I
70 LET S = S/10
80 FOR I = 1 TO 10
90 LET D = D + (V(I) -S) ↑ 2   (Cálculo da variância)
100 NEXT I
110 LET D = D/9
120 PRINT "MÉDIA ARITMÉTICA IGUAL A", S
130 PRINT "QUADRADO DA DISPERSÃO MÉDIA IGUAL A", D
140 DATA . . . ., . . . ., . . . ., . . . .
150 DATA . . . ., . . . ., . . . ., . . . .
160 END
```

7.1.3 Soma (ou subtração) de duas matrizes n x m

$$\begin{bmatrix} a_{11} & a_{12} & \dots \\ a_{21} & a_{22} & \dots \\ \dots & & \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots \\ b_{21} & b_{22} & \dots \\ \dots & & \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \dots \\ c_{21} & c_{22} & \dots \\ \dots & & \end{bmatrix}$$

```
NEW
10 DIM A(20,20), B(20,20), C(20,20)     (Tamanho máximo das matri-
20 INPUT N,M                             zes)
30 FOR I = 1 TO N
40 FOR J = 1 TO M
50 INPUT A (I, J), B (I, J)              (ou READ com DATA)
60 LET C(I,J) = A(I,J) + B(I,J)
70 PRINT C (I,J)
80 NEXT J
90 NEXT I
100 END
```

7.1.4 Multiplicação de duas matrizes

Para multiplicar uma matriz A = (a_{ij}) de tamanho m x p por outra matriz B = (b_{ij}) de tamanho p x n usamos a seguinte regra:

"O elemento cij da matriz produto C = (c_{ij}) é obtido multiplicando-se termo a termo, a linha i da matriz A e a coluna j da matriz B, somando-se os resultados obtidos."

O número de colunas de A deve ser igual ao húmero de colunas de B. Então temos:

$$\begin{bmatrix} a_{11} & \cdots & a_{1p} \\ & \cdots & \\ \boxed{a_{i1} \quad a_{i2} \cdots a_{ip}} \\ a_{m1} & a_{m_2} \cdots & a_{mp} \end{bmatrix} \times \begin{bmatrix} b_{11} & \cdots & b_{1j} & \cdots & b_{1n} \\ & & & & \\ b_{p1} & & b_{pj} & \cdots & b_{pn} \end{bmatrix} \times \begin{bmatrix} c_{11} & & c_{1n} \\ & \boxed{c_{ij}} & \\ c_{m1} & \cdots & c_{mn} \end{bmatrix}$$

$$\text{e } c_{ij} = a_{i1} b_{1j} + a_{i2} b_{2j} + \ldots + a_{ip} b_{pj} = \sum_{k=1}^{p} a_{ik} \cdot b_{kj} \quad (i = 1, 2, \ldots m \text{ e } j = 1, 2, \ldots n)$$

A matriz C terá tamanho m x n.

Escrever o respectivo programa em BASIC.

```
NEW
10 DIM A (20,20), B (20,20), C (20,20)      (Define tamanho máximo das matrizes)
20 READ M, P, N                             (Ler tamanho das matrizes)
30 FOR I = 1 TO M  }
40 FOR J = 1 TO P  }
50 READ A (I, J)   }                        (Leitura da matriz A(I,J))
60 NEXT J          }
70 NEXT I          }
80 FOR I = 1 TO P  }
90 FOR J = 1 TO N  }
100 READ B (I,J)   }                        (Leitura de B (I,J))
110 NEXT J         }
120 NEXT I         }
125 PRINT "MATRIZ PRODUTO C (I,J)"
130 FOR I = 1 TO M
140 FOR J = 1 TO N
150 FOR K = 1 TO P
```

```
160 LET C (I,J) = C (I,J) + A (I,K) * B (K,J)
170 NEXT K
180 PRINT C (I,J);          (Coloca C (I,J) na área de impres-
190 NEXT J                  são para imprimir linha I)
200 PRINT "b̸"               (Imprime uma linha I completa
210 NEXT I                  de C(I,J))
220 DATA 3, 4, 2            (Valores M, P e N)
230 DATA 4, 1, 5, 2
240 DATA 0, 2, 3, 0         (Matriz A de tamanho 3x4)
250 DATA 3, 0, 4, 0
260 DATA 2, 1.5             (Matriz B de tamanho 4x2)
270 DATA 3,2
280 DATA 1, 1
290 DATA 4, 2
300 END
```

```
RUN
MATRIZ    PRODUTO   C(I, J)
    24         17
     9          7
    10        8.5
READY
```

7.2 EXERCÍCIOS DE APLICAÇÃO

1. Dizer o que é um vetor e como é representado em Basic? Dar exemplos.
2. Dizer o que é uma matriz e como é representada em Basic? Dar exemplos.
3. Para que serve o comando **DIM**?
4. O que acontece se, no programa Basic, usarmos comandos com variável A (I,J) dimensionada, mas sem o uso do comando DIM?

Por exemplo:

```
05 INPUT N
10 FOR I = 1 TO N
20 FOR J = 1 TO N
30 INPUT A (I, J)
40 PRINT A (I, J)
50 NEXT J
60 NEXT I
70 END
```

5. Escreva um programa Basic que calcule a soma dos elementos da diagonal de uma matriz, isto é,

$$A\ (I, J) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

Soma dos elementos da diagonal $= a_{11} + a_{22} + \dots + a_{nn}$

6. Escrever um programa Basic que calcule o produto de uma matriz A (I, J) com um vetor X (J).
(Sugestão: adaptar o exercício de multiplicação de matrizes.)

Processamento Comercial de Dados Usando Basic: PRINT USING e Operação com Strings 8

8.1 PROCESSAMENTO COMERCIAL DE DADOS USANDO BASIC

O uso do Basic em aplicações comerciais envolve recursos adicionais dessa linguagem e que são:

- manipulação de *strings* ou seqüência de carateres;
- manipulação de arquivo de dados em disco magnético.

A grande dificuldade reside no fato que justamente esses dois recursos não são padronizados nas diferentes versões da linguagem Basic, variando grandemente em sua definição e capacidade de uso.

Sempre há a necessidade de estudar detalhadamente os recursos de manipulação de *strings* e arquivos, através de manuais de utilização de cada linguagem Basic que se acha em operação.

Apresentaremos idéias básicas sobre operações envolvendo seqüência de caracteres e comando **PRINT USING** que permite a formatação ou especificação dos dados de saída de modo a formar resultados apresentáveis em documentos ou relatórios comerciais. (Fig. 8.1) O uso de arquivo de disco em alguns microcomputadores está na Apêndice A.

8.2 DEFINIÇÃO DE VARIÁVEIS QUE CONTÊM *STRINGS* DE CARACTERES

Já sabemos que uma variável que contém seqüência de até 525 caracteres é definida através de um nome seguido pelo sinal "$", isto é, A$, B$, B1$, etc.

Exemplo:

```
10 A$ = "NOME DO ALUNO"
20 PRINT A$, "BOM"
```

Resultado:

```
NOME DO ALUNO BOM
```

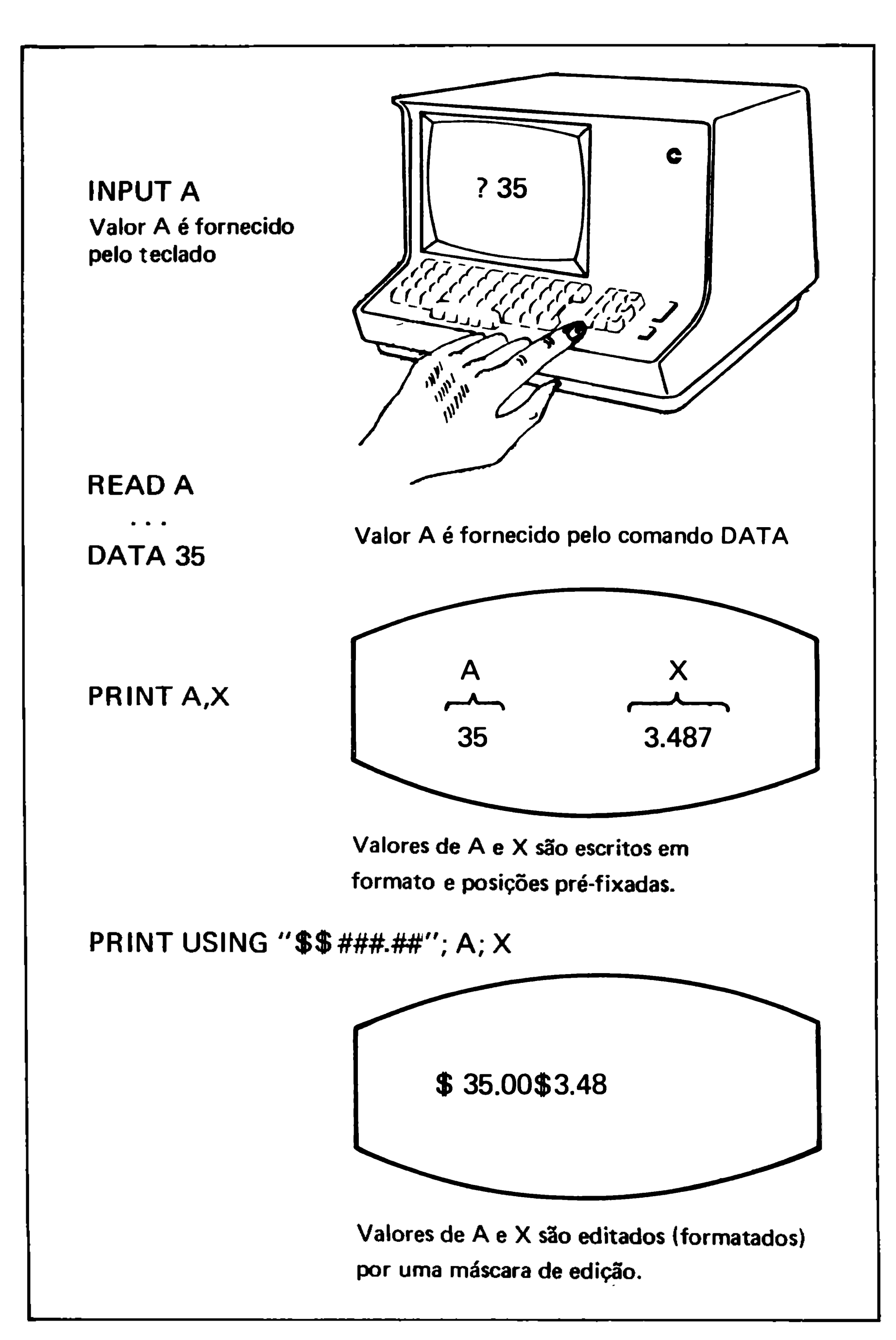


 Figura 8.1. *Formas diferentes de ler e escrever*

8.3 OPERAÇÕES BÁSICAS COM *STRINGS*

Exemplos:

a) Concatenação ou união de dois *strings*

Exemplo:

```
10 A$ = "ANO"
20 B$ = "BOM"
30 PRINT A$ + B$
```

Resultado:

```
ANO BOM
```

b) Comparação de *strings*

Exemplo:

```
30 IF Z$ = "MAXIMO" THEN A = A + 1
```

c) Conversão de *string* em valor numérico: função VAL(X$)

Exemplo:

10 X$ = "123"	• *string* 123
20 X = VAL (X$)	• X conterá valor inteiro 123

Outras operações com *strings* podem ser efetuadas através de funções apropriadas apresentadas no fim deste capítulo.

8.4 DEFINIÇÃO DE TABELAS (ARRAYS) DE *STRINGS*

Quando queremos trabalhar com um conjunto (tabela ou matriz) de nomes ou *strings,* podemos usar a seguinte definição:

```
10 DIM A1$ (11)
```

que define uma tabela A1$ de doze (de 0 a 11) *strings* distintos.

Exemplo:

```
   10 DIM A1$ (11)
   20 A1$(0) = "JANEIRO"
   30 A1$(1) = "FEVEREIRO"
   40 A1$(2) = "MARÇO"
   . . .
  120 A1$ (10) = "NOVEMBRO"
  130 A1$(11) = "DEZEMBRO"
   . . .
```

Observação: Em algumas versões Basic, entretanto, **DIM A1$(11)** define onze *strings:* de **A1$(1)** a **A1$(11)**, o que parece mais coerente. A definição de matrizes ou *strings* de duas dimensões também é possível através do comando do tipo:

10 DIM A2$(4, 3)

onde **A2$** pode conter os dados:

(Tempo no período manhã, tarde, noite)

BOM	BOM	BOM
BOM	NUBLADO	CHUVOSO
CHUVOSO	CHUVOSO	CHUVOSO
BOM	BOM	BOM

Exemplos:

```
A (1, 1) = "BOM"
A (2, 3) = "CHUVOSO"
```

Nota: Deve-se verificar sempre o uso correto de tabelas com *string* na versão Basic, em uso, pois em algumas versões A2$ (2, 3) pode significar os caracteres das posições 2 a 3 do *string* A2$, em vez do elemento (2, 3) da matriz A2$. . .

Devemos lembrar também que o uso de tabelas (ARRAYS) de *strings* sempre ocupam grande espaço na memória. Por isso *arrays* de *strings* devem ser utilizados com bastante critério.

8.5 PREPARAÇÃO (EDIÇÃO E FORMATAÇÃO) DE DADOS DE SAÍDAS

8.5.1 Impressão com formato fixo dos dados

```
10 V1 = 10
20 V2 = 20
30 V3 = 30
40 V4 = 40
50 PRINT V1, V2
60 PRINT V3, V4
70 END
```

Resultado:

```
RUN
10    20
30    40
```

onde os dados são separados por número fixo de espaço.

8.5.2 Uso da função TAB para controlar espaço entre os dados

Podemos definir o espaçamento que queremos usando **TAB** e ";".

```
20 PRINT TAB (15); V1; TAB (15); V2
30 PRINT TAB (15); V3; TAB (15); V4
...
```

Resultado:

(15 espaços) **10** (15 espaços) **20**

(15 espaços) **30** (15 espaços) **40**

8.5.3 Comando PRINT USING para editar e preparar dados de saída

Este comando possibilita editar os valores de saída através de colocação de sinais "$", "*", o ponto e vírgula em qualquer parte do valor numérico. A edição é feita através de uma "máscara" de edição formada por esses sinais e pelo sinal " # " colocado na posição onde deve ocorrer um dígito numérico.

Os sinais possíveis de edição são:

- **#** : indica que nesta posição deve aparecer um dígito numérico.
- (ponto): idem, um ponto.
- **,** (vírgula) : idem, uma vírgula.
- **$** : idem, um sinal "$".
- **$$** : sinal "$" flutuante, isto é, coloca um sinal $ à esquerda do algarismo ou dígito mais significativo.
- ****** : preenche todas as posições à esquerda do dígito significativo com o sinal "* ".
- ****$** : coloca asterisco antes do sinal $ flutuante.

A máscara de edição pode aparecer dentro do próprio comando **PRINT USING**, ou ser colocada em uma variável qualquer, em forma de *strings* de caracteres.

No exercício a) apresentamos esses dois casos que são equivalentes.

Exemplo:

a)
```
10 X1 = 4342.23
20 PRINT USING "$ #,###.##"; X1
```
Resultado: **$ 4,342.23**

ou

```
10 X1 = 4342.23
15 U$ = "$#,###.##"
20 PRINT USING U$; X1
```

b)
```
30 X3 = 98.7654
40 PRINT USING "VALOR = **$ ###.##"; X3
```
Resultado: **VALOR = **$98.76**

c)
```
10 X1 = 1.234
20 X2 = 25.71
30 X3 = 345.8
40 PRINT USING "$###.## $###.## $###.##", X1; X2; X3
```
Resultado: **$1.23 $25.71 $345.80**

Como ocorre a edição de dados

A edição dos valores através das máscaras do comando PRINT USING ocorre do seguinte modo:

Exemplos:

---

a) Valor: **4 342.23**

Máscara: **$#,###.##**

Resultado editado: **$ 4,342.23**

---

b) Valor: **98.7654**

Máscara: **VALOR = **$# # #.# #**

Resultado: **VALOR = **$98.76**

A posição do ponto decimal

A posição do ponto decimal do valor (no Brasil usa-se vírgula decimal) fica automaticamente ajustada com o ponto decimal da máscara.

No exemplo b) como houve menos dígito, na parte inteira do que o previsto, os sinais (**$) foram deslocados ou flutuados para a direita.

Um exercício interessante seria colocar uma "vírgula decimal" em vez do ponto decimal. Na maioria do Basic parece ser impossível editar diretamente a vírgula nessa posição. Uma alternativa seria transformar o valor numérico em valor inteiro, multiplicando-o por 1000 e desprezando as demais casas decimais usando a função **INT(X)**, após o que efetua-se a edição da vírgula entre a segunda e terceira casa (**INT(X)** separa maior inteiro contido em X).

Exemplo:

```
10 X = INT (X * 1000)
20 PRINT USING "$# # #,# #"; X
```

8.5.4 Uso do PRINT USING

Exemplo:

```
10 A = 10.356,B = 101.4589,C = 1001.037,D = 1.097
20 PRINT USING "###.###"; A, B, C, D
30 END
```

```
RUN

10.356
101.459          (arredondamento)
1001.037         (mensagem de erro, pois contém mais de
                 três algarismos na parte inteira)
1.097

READY
```

Observação: Comando número 10 contém várias operações aritméticas.

8.6 FUNÇÕES QUE MANIPULAM *STRINGS* DE CARACTERES

8.6.1 CHR$(X) – Converte valor numérico X em caractere ASCII

Faz a conversão de um valor numérico ou resultado de uma expressão aritmética X, para o código ASCII correspondente. (Ver Apêndice B). Essa função pode ser utilizada como recurso especial, durante o processamento de um programa, em operações como:

1. Dar um retorno de carro
2. Dar um *line feed*
3. Limpar a tela do vídeo
4. Tocar um sinal de alarme para indicação de erro em um programa.

Exemplos:

Consideremos os seguintes códigos de caracteres ASCII e respectivos valores numéricos:

Caráter ASCII	Código Numérico
LF *(line feed)*	**10**
CR *(carriage return)*	**13**
SP *(space)*	**32**
F *(letra F)*	**70**

Então:

```
10 PRINT CHR$(10)                        – executa a operação line feed
20 PRINT CHR$(13)                        – executa a operação carriage
                                           return
30 PRINT CHR$(70); CHR$ (32); CHR$(70)   – imprime os caracteres:
                                           F b F (b = espaço)
```

8.6.2 ASC (N$): Converte caractere ASCII em valor numérico

Fornece o valor numérico correspondente ao código ASCII do primeiro caractere do *string* **N$**. Na realidade é a função inversa da **CHR$(N)**. Caso o *string* **N$** não seja definido, o interpretador Basic acusará uma mensagem de erro.

Exemplo:

```
10 A$ = "BASIC"
20 PRINT ASC(A$)
30 END
RUN
```

Resultado:

```
66                (escreveu o valor 66 correspondente ao
                  caractere B no código ASCII)

READY
```

8.6.3 LEFT$ (N$, X) – Fornece X caracteres mais à esquerda do *string*

Esta função separa os X caracteres mais à esquerda (ou X primeiros caracteres) do *string* **N$**.

Exemplo:

```
10 N$ = "BASIC"
20 PRINT LEFT$ (N$, 3)
30 END
```

```
RUN
BAS
READY
```

8.6.4 RIGHT$ (N$,X) – Fornece X caracteres mais à direita do *string*

Essa função é simétrica a **LEFT$**, pois fornece os X últimos caracteres (ou caracteres mais à direita) de um *string* **N$**.

Exemplo:

```
10 N$ = "BASIC"
20 PRINT RIGHT$(N$, 3)
30 END
```

```
RUN
SIC
READY
```

8.6.5 MID (N$, X, Y): Fornece caracteres do meio do *string*

Essa função fornece os Y caracteres de *string* **N$** contados a partir da posição X em diante.

Exemplo:

```
10 N$ = "FUNCAO"
20 PRINT MID$ (N$, 3, 3)
30 END
```

```
RUN
NCA
READY
```

8.6.6 STR$ (X) – Transforma valor numérico em *string*

Transforma os algarismos de um valor X em um *string*, onde X pode ser um número ou uma variável numérica. Depois que a função **STR$(X)** é executada, os algarismos do *string* obtido não podem participar de operações aritméticas, apesar de representar um número. Esta função é diferente de **CHR$(X)**, onde o valor numérico X é convertido em um único código ASCII correspondente, ao passo que, em **STR$(X)**, cada algarismo, ponto decimal e o sinal que formam o valor X, são convertidos em seqüência *(string)* de caracteres ASCII.

Exemplo:

```
10 X = 21.7          (valor numérico 21.7)
20 B$ = STR$(X)      (string de caractere "21.7")
30 PRINT X + 1, B$
40 END
```

```
RUN
22.7      21.7      (seqüência de algarismos transformada para
                     string, não podendo sofrer operações aritmé-
                     ticas)
```

8.6.7 VAL (N$) – Transforma *string* em valor numérico

É a função inversa da **STR$(X)**, ou seja, **VAL(N$)** é utilizada para voltar a dar a característica de um número a um *string*, transformado pela função **STR$**.

Com essas duas funções, podemos transformar um nº em *string*, executar a operação de *string* desejada e novamente transformá-lo em número.

8.6.8 Aplicação Comercial – Edição de sinal $ e acerto da posição da casa decimal

Exemplo:

Na seção 2.5, Listagem de relatório do PNB (Produto Nacional Bruto), a posição das casas decimais dos valores não estão alinhados e os valores não contém o sinal **$**.

Podemos definir a máscara.

```
225 LET M$ = "$# # # #.# #"
```

e transformar todos os comandos **PRINT** em **PRINT USING** e assim acertar a posição das casas decimais.

Exemplo:

```
240 PRINT USING "PRODUCAO INDUSTRIAL         ", M$;D
250 PRINT USING "PRODUCAO AGRO-PASTORIL   ";M$;N
```

(etc.)

e teremos o seguinte relatório:

```
PRODUCAO INDUSTRIAL          $ 179.30
PRODUCAO INDUSTRIAL          $ 251.50
. . .
INVESTIMENTO NO PAIS         $   53.00
. . .
DEPRECIACAO                  $   53.30
PRODUTO NACIONAL BRUTO       $ 949.60
```

8.7 EXERCÍCIOS DE APLICAÇÃO

1. Qual a diferença entre os comandos **PRINT** e **PRINT USING**?
2. Quais são os sinais de edição do comando **PRINT USING**? Como deve ser usado cada sinal?
3. Qual será o resultado escrito por cada comando **PRINT USING**?

```
10 LET A = 3457.847
20 LET A$="$$##,###.##"
30 PRINT USING A$; A
40 LET B$ = "$###,###,###.####"
50 PRINT USING B$; A
60 LET C$ = "**#########"
70 PRINT USING C$; A
80 END
```

4. Que faz a função **TAB(X)**? Dar exemplos.
5. Dar exemplos de comandos e resultados da operação das seguintes funções:

   CHR$(X), ASC(A$), LEFT$(A$, X), RIGHT (A$, X),

   MID (A$, X, Y), STR$(X) e **VAL(A$).**

6. Se

   A$ = "VALOR MONETARIO"

   X = 5

   Y = 3

   dizer qual o resultado da aplicação de cada função que manipula *string* **CHR$, ASC$, LEFT$** etc., citados no exercício anterior.

Sub-rotinas e Funções Definidas pelo Usuário: Comandos GOSUB, RETURN, ON e DEF



9.1 DEFINIÇÃO E USO DE SUB-ROTINAS

Uma sub-rotina é um trecho do programa que é executado quantas vezes se queira, através de um comando chamado **GOSUB**. Quando uma mesma operação é executada várias vezes no mesmo programa, é conveniente separá-la como sub-rotina e executar através do comando **GOSUB**.

GOSUB n	Desvia para o comando de número *n* como se fosse uma sub-rotina e retorna quando encontrar o comando **RETURN.**
RETURN	Retorno do desvio devido ao comando **GOSUB.**

Exemplo:

```
10 . . .                                   }
20                                         }
30 GOSUB 500    (deseja-se calcular        }  programa principal
35 . . .         alguma coisa)             }
   . . .
500                                        }
510                                        }
. . .                                      }  Sub-rotina que calcula operações
550 RETURN                                 }  solicitadas por GOSUB 500
600 END                                    }
```

Observação: Após a execução da sub-rotina 500, o programa retorna para a linha 35 que está imediatamente após o **GOSUB 500.**

9.2 COMANDO DE ESCOLHA ALTERNATIVA DE GOSUB OU GOTO

ON Permite a escolha de um dos vários comandos **GOTO** ou **GOSUB.**

Exemplo de **GOTO**:

Para valor de X igual a 1, 2 ou 3, desvie para comandos 50, 20 e 30, respectivamente.

```
05 ON X GO TO 50, 20, 30
20 . . .                          (desvio se X= 2)

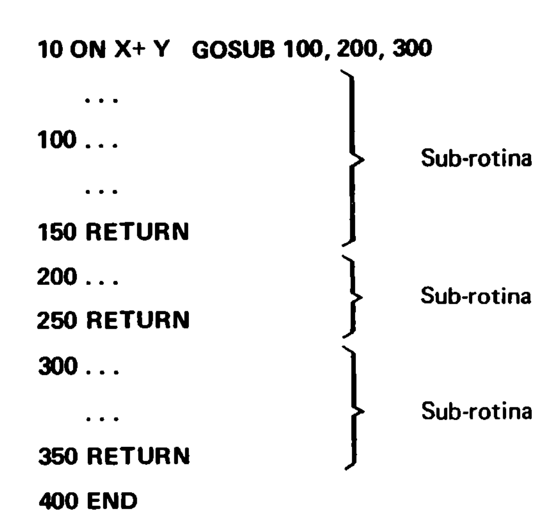
30 . . .                          (desvio se X= 3)

50 . . .                          (desvio de X= 1)
```

Observação: O desvio é feito para o número do comando que está na posição indicada por X.

Exemplo de **GOSUB:**

Se o valor da expressão X+Y for 1, 2 ou 3, desvie para sub-rotina 100, 200 ou 300, respectivamente.



9.3 FUNÇÃO MATEMÁTICA **DEF FNa(X)** DEFINIDA PELO USUÁRIO

DEF FNa(X)	Define uma equação matemática **FNa(X)** que o usuário pode usar nos comandos **LET, PRINT, IF** etc. A terceira letra *a* deve ser fornecida pelo usuário para identificar uma função da outra.

Exemplo:

10 DEF FNA(S) = 3 *X+2	(define $f_1(x) = 3x + 2$)
20 DEF FNB(X) = (X * *2+1)	(define $f_2(x) = x^2 + 1$)
30 PRINT FNA(5)	(para X = 5, escreve $f_1(5) = 3 \times 5 + 2 = 17$)
40 PRINT FNB(3)	(para X = 3, escreve $f_2(3) = 3 \times 3 + 1 = 10$)
50 LET W = FNA(2) *4	(calcula W = (3 x 2 + 2) x4)

Nota: **FNA(5), FNB(3)** e **FNA(2)** não são confundidos como variáveis dimensionadas, pois no Basic oficial as variáveis dimensionadas possuem uma só letra no nome. (Exemplo: F(2), N(3), etc.). Em Basic não padronizado, que permite 3 letras como nome de variáveis, é preciso consultar o manual do microcomputador.

9.4 DIFERENÇA ENTRE GOSUB E DEF FNa(X)

O comando **DEF FNa(X)** só pode calcular um único resultado numérico, pois é formado por uma única equação matemática.

A sub-rotina, chamada por **GOSUB**, pode ser formada por qualquer número de comandos, e portanto calcular mais de um resultado numérico.

Exemplo:

Para calcular a função:

$f(x) = x^2 + 2$	para $0 < x < 1$
$f(x) = 0$	para demais valores de x devemos usar a sub-rotina com **GOSUB**, pois f(x) possui mais de uma alternativa de cálculo.

```
10 LET X = 2
20 GOSUB 500          (função F(x) será 0)
23 PRINT F
25 LET X = 0.2
30 GOSUB 500          (função F(x) será 2.04)
35 PRINT F
40 GO TO 600
500 REM FUNCAO F(X)
510 LET F = 0
520 IF X <= 0 THEN RETURN
530 IF X >= 1 THEN RETURN
540 LET F = X*X+2
550 RETURN
600 END
```

Nota: No exemplo acima *não* é preciso usar variável na forma F(x) que seria interpretada como variável dimensionada, o que não é o caso. Notar que a forma **F(x)** só foi usada como comentário em **500 REM FUNCAO F(x).**

Exemplo de comparação de rotinas com e sem uso de **GOSUB-RETURN**

> Calcular os valores de
>
> $$S = \frac{x^k}{k!} + (x!) \qquad \text{e} \qquad T = (k/2)!$$
>
> para valores de X e k dados (k = par).

Programa Basic:

SEM GOSUB-RETURN	COM GOSUB-RETURN	
NEW 10 INPUT X, K	NEW 10 INPUT X, K	
20 LET F1 = 1 30 FOR I = 1 TO K 40 LET F1 = F1 * I 50 NEXT I	20 LET K1 = K 30 GOSUB 120	(fatorial de K)
60 LET S = (X↑ K)/F1	40 LET S = (X ↑ K)/F1	
70 LET F1 = 1 80 FOR I = 1 TO X 90 LET F1 = F1 * I 100 NEXT I	50 LET K1 = X 60 GOSUB 120	(fatorial de X)
110 LET S = S + F1 120 LET T = 1 130 FOR I = 1 TO K/2 140 LET T = T *I 150 NEXT I	70 LET S = S + F1 80 LET K1 = K/2 90 GOSUB 120	(fatorial de K/2)

```
160 PRINT S,T
170 END
```

```
100 PRINT S, F1
110 END
120 LET F1 = 1
130 FOR I = 1 TO K1
140 LET F1 = F1 * I
150 NEXT I
160 RETURN
170 END
```

(sub-rotina para fatorial)

Observação:

Os comandos **GOSUB-RETURN** tornam o programa principal mais curto e elegante, isto é, o programa torna-se melhor estruturado. O tempo gasto na execução é sempre o mesmo.

9.5 EXERCÍCIOS DE APLICAÇÃO

1) O que faz o comando **GOSUB** e **RETURN**? Dar exemplos.

2) O que faz o comando **DEF**? Dar exemplos.

3) Definir e usar as seguintes funções:

3a) $f(x) = kx$ (k e x dados)

3b) $f(x) = 2x + x^2$

3c) $f(x) = k$ para $0 < x < 1$

$f(x) = x$ para $x \leqslant 0$

$f(x) = x^2 + k$ para $x \geqslant 1$

(k = constante)

4. Transformar o Exercício da seção 9.4 com **GOSUB** em programa equivalente, usando **DEF FNa(x)**

(Sugestão: usar dois comandos **DEF FNa(X)**)

Principais Funções Fornecidas pelo Basic 10

10.1 RESUMO DAS FUNÇÕES

O Basic pode fornecer as seguintes funções prontas para o uso:

Funções aritméticas

Nome	*Operação*	*Exemplo*
ABS(X)	valor absoluto de X	**LET Y = ABS(X+3)**
LOG(X) ou **LN(X)**	logaritmo natural de X	**LET Y = LOG(X)**
EXP(X)	valor da exponencial e^X	**LET Y = EXP(3.1415)**
SQR(X)	raiz quadrada de X	**LET Y = SQR(2)**
SGN(X)	valor 1 se $X > 0$, 0 se $X = 0$ e -1 se $X < 0$.	**LET SGN(X*3)**
INT(X)	maior inteiro contido em X	**LET Y = INT(X)**
(*)FIX(X)	parte inteira de X	**LET Y = FIX(X)**
RND(X)	valor aleatório entre 0 e 1.	**LET Y = RND(X)**
(*)CDBL(X)	converte X para precisão dupla	
(*) CSNG(X)	converte X para precisão simples	

Funções trigonométricas

SIN(X) –	seno de X (X em radianos)	**LET Y = SIN(3.1415)**
COS(X) –	cosseno de X (X em radianos)	**LET Y = COS(3.1415)**
TAN(X) –	tangente de X (X em radianos)	**LET Y = TAN(X)**

Funções trigonométricas inversas

ATN(X) –	arco-tangente de X	**LET Z = ATAN(1)**
(*)ACS(X) –	arco-cosseno de X	
(*)ASN(X) –	arco-seno de X	

Funções de controle de impressão

TAB(X) –	posiciona impressão na coluna X	**PRINT TAB (22)**

Funções que manipulam strings de caracteres (ver exemplos no capítulo sobre **PRINT USING**)

CHR$(X) – converte valor X em caractere ASCII

ASC(N$) – converte primeiro caractere do *string* em valor numérico

(*) LEFT$(N$,X) – fornece X caracteres mais à esquerda do *string* **N$**

(*) RIGHT(N$,X) – fornece X caracteres mais à direita do *string* **N$**

(*) MID$(N$,X,Y) – fornece Y caracteres a partir da posição X de **N$**

(*) STR$(X) – transforma valor numérico X em *string* de caracteres

(*)VAL$(N$) – transforma *string* **N$** em valor numérico.

Exemplos:

```
PRINT CHR$ (32)
PRINT ASC(A$)
```

Observação:

a) (*) – pode não estar disponível em algumas versões do Basic.

b) – O argumento X da função pode ser:
   - uma variável: X, Z, A1, A(3)
   - uma constante: 3.1415, 1, -100
   - expressão aritmética: X + 3 * Y, 3.1415 + X

10.2 FUNÇÕES ARITMÉTICAS

10.2.1 **ABS(X)**: Valor absoluto de X

Fornece o valor absoluto (módulo) do argumento X que está entre parânteses.

Se o argumento for um nº positivo ($X \geqslant 0$) a função retornará com o próprio valor do argumento.

Se o argumento for um nº negativo ($X < 0$) a função retornará com o valor do módulo de X.

Exemplos:

```
10 PRINT ABS(-10)
20 PRINT ABS(20)
30 X = -50
40 PRINT ABS(X)
50 PRINT ABS(9 *(-4))
60 PRINT ABS (X* (-3))
70 END
```

Resultados:

```
RUN
10 (linha 10)
20 (linha 20)
50 (linha 40)
36 (linha 50)
150 (linha 60)
```

10.2.2 LOG(X) (ou LN(X)) – Logaritmo Natural

Fornece o logaritmo natural (base e) do argumento X (X > 0)

Exemplos:

```
10 PRINT LOG(10)
20 PRINT LOG(3)
30 X = -3
40 PRINT LOG(X*((-2))
50 END
```

Resultado:

```
RUN
2.302585
1.098612
1.791759
```

10.2.3 Mudança de Base do Logaritmo

Para se determinar o logaritmo de um número em uma outra base A, deve-se proceder da seguinte forma:

$$LOG_A(X) = LOG_e(X)/LOG_e(A)$$

$$BASE = A$$

Exemplo:

Suponha que desejemos calcular o logaritmo de 2 na base 10, ou seja, $LOG_{10}(2)$.

Fazemos o programa abaixo:

```
10 X = 2
20 A = 10
30 PRINT LOG(X)/LOG(10)
40 END
```

Resultado:

```
RUN
0.301030
READY
```

10.2.4 EXP(X) – Exponencial e^X

Calcula o valor de e^X. É a função inversa do logaritmo natural, ou seja, calcula o inverso da função **LOG(X)**.

Portanto:

EXP(LOG(X)) vai valcular o próprio valor x.

Exemplo:

```
10 PRINT EXP(2.302585)
20 PRINT EXP(1.098612)
30 END
```

Resultado:

```
RUN
10
3
READY
```

10.2.5 SQR(X) – Raiz Quadrada de X

Fornece a raiz quadrada do argumento X.

O argumento deve ser um número positivo maior ou igual a zero.

Caso contrário será indicada uma mensagem de erro.

Exemplos:

```
10 PRINT SQR(4)
20 X = 3
30 Y = 27
40 PRINT SQR(X),SQR(Y),SQR(X*Y)
50 END
```

```
RUN

2
1.732051          5.196152          9
READY
```

10.2.6 SGN(X) – Sinal de X

Esta função faz o teste do sinal do argumento X.

Fornece, como resultado, os valores abaixo relacionados:

1 se $X > 0$
0 se $X = 0$
- 1 se $X < 0$

Exemplos:

```
10 PRINT SGN(10)
20 PRINT SGN (-15)
30 X = 13
40 Y = - 3
50 PRINT SGN(X), SGN(Y), SGN(X* Y)
60 END
```

```
RUN
1
-1
1          -1          -1
```

10.2.7 INT(X) – Maior inteiro contido em X

Esta função fornece o maior valor inteiro menor ou igual a X.

Exemplo:

```
READY
10 PRINT INT (75.78), INT (-11.13)
20 END
```

```
RUN
75              -12
```

10.2.8 FIX(X) – Parte inteira de X

Esta função fornece a parte inteira de X.

A função **FIX(X)** é considerada equivalente à **SGN(X) X INT(ABS(X))**.

A diferença principal entre a função **FIX** e **INT** é que **FIX(X)** fornece o inteiro seguinte(e maior)ao valor X quando este é negativo e não o maior inteiro contido em X, como acontece se X é positivo.

Exemplo:

```
READY
10 PRINT FIX(77.82)
20 END
```

```
RUN
77
READY
```

```
10 PRINT FIX (-77.82)
20 END
```

```
RUN
-77
READY
```

10.2.9 RND(X) — Valor aleatório entre 0 e 1

Esta função gera, por sorteio, um valor aleatório eqüiprovável entre 0 e 1. Na teoria da probabilidade, dizemos que **RND(X)** gera um valor com distribuição uniforme.

Esta função é diferente das demais, no sentido de que basta fornecer um valor inicial X (entre 0 e 1) para termos, automaticamente, o valor aleatório eqüiprovável, a cada utilização da função **RND(X)**.

O valor inicial X (chamado "semente") serve para fornecer uma seqüência aleatória de valores entre 0 e 1, à medida que usarmos **RND(X)**. Para outro valor inicial X, teremos uma seqüência diferente. A função **RND(X)** é usada no programa de simulação e de jogos de azar.

Exemplos:

10 LET X = 0.35	(valor inicial X = 0.35)
15 FOR I = 1 TO 5	
20 LET Y = RND(X)	fornece uma seqüência de cinco valores
30 PRINT Y	aleatórios; por exemplo: 0.689, 0.234, 0.543,
40 NEXT I	0.378, 0.607.
50 FOR I = 1 TO 3	
60 LET Z = RND(.77)	(valor inicial X = 0.77)
70 PRINT Z	fornece outra seqüência de três valores aleató-
80 NEXT I	rios; por exemplo: 0.971, 0.734, 0.451
90 END	

10.2.10 CDBL(X) – Converte X para precisão dupla

Faz a conversão de uma variável numérica ou de uma expressão aritmética para um número de dupla precisão.

Exemplo:

```
10 X = 454.67
20 PRINT CDBL(A)
30 END
```

```
RUN
454.67001234277344
READY
```

10.2.11 CSNG(X) – Converte X para precisão simples

Faz a conversão de uma variável numérica ou de uma expressão aritmética de dupla precisão em um número de simples precisão.

10.3 FUNÇÕES TRIGONOMÉTRICAS

10.3.1 SIN(X) – Seno de X

Fornece o valor do seno do argumento X que deve estar expresso em radianos.

Exemplos:

```
10 PRINT SIN(0.523599)
20 PRINT SIN(0.785398)
30 X = 1.047198
40 Y = 1.570796
50 PRINT SIN(X), SIN(Y)
60 END
```

```
RUN
0.5
0.707107
0.866026               1.0000
```

Nota: A precisão do valor obtido pode variar de versão para versão.

10.3.2 COS(X) – Co-seno de X

Fornece o valor do co-seno do argumento X que deve ser expresso em radianos.

Exemplos:

```
10 PRINT COS(1.047198)
20 PRINT COS(0.523599)
30 X = 0.785398
50 PRINT COS(X)
60 END
```

```
RUN
0.499833
0.866026
0.707107
```

10.3.3 TAN(X) – Tangente de X

Fornece o valor da tangente do argumento X que deve estar expresso em radianos.

Exemplos:

```
10 PRINT TAN(0.523599)
20 PRINT TAN(0.785398)
30 X = 1.047198
40 Y = 1.570796
50 PRINT TAN(X), TAN(Y)
60 END
```

```
RUN
0.577350
1.00000
1.732051          (*)
```

Observação: Não está definido o valor da tangente de $\pi/2$ ou 1.570796.

10.4 FUNÇÕES TRIGONOMÉTRICAS INVERSAS

10.4.1 ATN(X) – Arco tangente de X

Fornece, em radianos, o valor do arco, cuja tangente é X. O valor do arco estará compreendido entre $-\pi/2$ a $+\pi/2$.

Exemplos:

```
10 PRINT ATN(1)
20 PRINT ATN(.577350)
30 X = 1.732051
40 Y = 3.732051
50 PRINT ATN(X), ATN(Y)
60 END
```

```
RUN

0.785398
0.523599
1.047198              1.308997
```

Para obter o valor em graus dos valores numéricos acima, basta multiplicar **ATN(X)** por **57.29580.**

Exemplo:

```
10 PRINT 57.2958 * ATN(1)
20 PRINT 57.2958 * ATN (.577350)
```

10.4.2 ACS(X) – Arco co-seno de X

Fornece, em radianos, o arco cujo co-seno é X.

Exemplos:

```
10 PRINT ACS(0.499833)
20 PRINT ACS(0.866026)
30 X = 0.707107
50 PRINT ACS(X)
60 END
```

```
RUN

1.047198
0.523599
0.785398
```

10.4.3 ASN(X) – Arco seno de X

Fornece o arco cujo seno é o valor do argumento X.

Exemplos:

```
10 PRINT ASN(0.5)
20 PRINT ASN(0.707107)
30 X = 0.866026
40 Y = 1
50 PRINT ASN(X), ASN(Y)
60 END
```

```
RUN
0.523599
0.785398
1.047198                1.570796
```

10.5 APLICAÇÕES

Exemplos de uso das funções trigonométricas:

> Escrever uma tabela de **SIN(X)** (seno) colocando os valores de X (em radianos) e **SIN(X)** dispostos em duas colunas de impressão. X varia de 0 a 3.455 radianos com incremento igual a π /10 (0.31415).

a) Programa usando função **SIN(X)**

```
10 PRINT "X–RD        SIN(X)        X–RD        SIN(X)"
20 PRINT " I––––––I        I––––––––––––I"
30 FOR X = 0 TO 1.884946 STEP 0.31415
40 LET Y = SIN(X)
50 LET X1 = X + 1.88485
60 LET Y1 = SIN(X1)
70 PRINT X, Y, X1, Y1
80 NEXT X
90 END
```

b) Programa usando sub-rotina própria **SEN(X)** e comando **GOSUB.**

```
0010 PRINT 'X–RD       SEN(X)       X–RD        SEN(X)"
0020 PRINT "I ———————————————I      I———————————I"
0030 FOR K = 0 TO 1.884946 STEP 0.3415927
0032 LET X = K
0035 GOSUB 1000
0040 LET X = X1
0050 LET X = X + 1.88485
0055 GOSUB 1000
0060 LET Y1 = X1
0070 PRINT K, Y, X, Y1
0080 NEXT K
0090 END
1000 REM SENO X
1005 REM X1 = SEN(X)
1015 REM
1020 LET N1 = 1
1025 LET N2 = -1
1030 LET X1 = 0.0
1035 LET X2 = 1.0                                  } sub-rotina SEN(X)
1040 LET X3 = X
1055 FOR N = 1 TO 15
1056 LET N1 = N1 * N
1060 LET X2 = X2 * X3
1070 IF N = (INT (N/2) * 2) THEN 1085
1075 LET N2 = INT (N2 * (-1))
1080 LET X1 = X1 + (X2 * INT (N2))/(INT (N1))
1085 NEXT N
1090 RETURN
```

Resultados:

```
#RUN
X-RD          SEN(X)          X-RD          SENX)
I-----------------I I-------------------I

0             0               1.88485       0.951089137
0.31415927    0.309016996     2.19900927    0.809079063
0.62831854    0.587785257     2.51316854    0.587870653
0.94247781    0.809016999     2.827332781   0.309117275
1.25663708    0.951056525     3.14148708    1.04816611E - 04
1.57079635    0.999999996     3.45564635    -0.308920425
```

c) Programa para traçar gráficos do seno

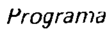
Traçar um gráfico da função **SIN(X)** para X variando de 0 a 2π .

Resolução: Como **SIN(X)** varia de -1 a +1, fixamos uma escala de -20 a 20 posições de impressão para **Y** = **INT(SIN(X) * 20)**

Assim:

SIN(X) = -1 corresponde à posição -20 do gráfico
SIN(X) = -0.5 corresponde à posição -10 do gráfico
SIN(X) = 0 corresponde à posição 0 do gráfico
SIN(X) = 0.5 corresponde à posição +10 do gráfico
SIN(X) = +1 corresponde à posição +20 do gráfico

etc.

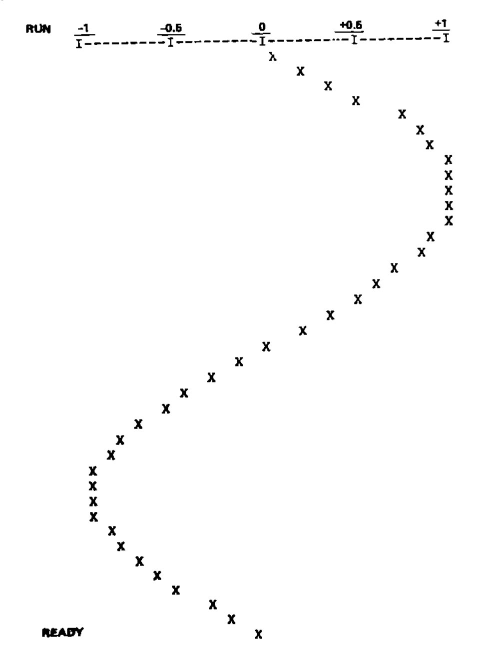


Programa

```
 10 PRINT "-1             -0.5        0      +0.5          +1"
 20 PRINT "I-----------I-------I-----I-----------I"
 30 FOR X = 0 TO 6.28318  STEP 0.15707
 40 LET Y = INT(SIN(X) * 20)
 50 FOR I = -20 TO Y
 60 PRINT "b̸"
 70 NEXT I
 80 PRINT "X"
 90 NEXT X
100 END
```

Observação: (b̸) — espaço em branco.

Resultado do exercício



10.6. EXERCÍCIOS DE APLICAÇÃO

1. Escrever, usando Basic, os valores de cada função deste capítulo no intervalo fornecido.

   Por exemplo: **Y = ABS(X)** para X = -50 a + 50

   Y = SQR(X) para X = -50 a +50

   etc.

2. Alterar o exemplo de **SIN(X)** para escrever uma tabela do **COS(X)** e **TAN(X)** para X de 0 até 3.455 radianos:

3. Traçar gráficos das funções cujos valores foram calculados no exercício 1.

   (Sugestão: modificar o exemplo de **SIN(X)**.

Aplicação: Cinco Fórmulas Diferentes para Obter o Valor de π 11

11.1 FÓRMULA CHINESA (CERCA DE 470 a.C)

"O valor de π foi expresso como número racional 355/113."

Nota: nº racional é a representação de um valor numérico como quociente de dois valores inteiros.

Resultado:

Com **PRINT 355/113**

temos **π = 3.1415929**

11.2 FÓRMULA DO INDIANO ARYABHATA (CERCA DE 510 a.C.)

"O valor de π foi expresso como número racional 62832/20000."

Resultado:

PRINT 62832/20000

π = 3.1416

* Adaptado de ***Mathematics teaching.*** New York, Harper &Row.

11.3 FÓRMULA DO MATEMÁTICO INGLÊS J. WALLIS (SÉCULO XVII)

"O valor π foi expresso como uma série de produtos de números racionais $\pi = 4 \times (2/3) \times (4/3) \times (4/5) \times (6/5) \times (6/7) \times (8/7) \times \ldots$ sempre com um número par de fatores após o 4."

Programa Basic:

```
 05 PRINT "QUAL O NUMERO N(PAR) DE FATORES DESEJADO?"
 10 INPUT X
 20 LET Y = INT (X/2)
 30 LET P = 4
 40 LET F = 2
 50 FOR I = 1 TO Y
 60 LET P = P * (F/(F + 1))
 70 LET P = P *((F + 2 )/ (F + 1))
 80 LET F = F + 2
 90 NEXT I
100 PRINT "VALOR DE PI PELA FORMULA DE WALLIS", P
110 END
```

Resultados:

Para

X = 2

Y = 1

P = 4

F = 2

P = 4 x (2/3 x (4/3))

valor π = **3.5555556**

Para

X = 4

Y = 2

P = 4

F = 2

I = 1: P = 4 x (2/3) x (4/3)

I = 2: P = Px (4/5) x (6/5)

valor π = **3.4133333333**

11.4 FÓRMULA DO MATEMÁTICO AUSTRÍACO STRASSNITZKY

$$\pi = 4 \times (\text{ARC TAN } (1/2) + \text{ARC TAN } (1/5) + \text{ARC TAN } (1/8)$$

Resultado:

100 LET P = 4 *(ATN(0.5) + ATN(0.2) + ATN(0.125))

valor $\pi = 4 \times (0.4636476+0.1973956+0.124355) = 3.1415928$

11.5 FÓRMULA DE LEIBNITZ (1674)

$$\pi = 4 *(1 -(1/3) + (1/5) - (1/7) + (1/9) - (1/11)...)$$

com o número de termos tendendo para infinito.

Programa Basic:

```
 10 PRINT "QUAL O NUMERO DE TERMOS"
 20 INPUT A
 25 LET A1 = 2 * A-1
 30 LET S = 1
 40 FOR I = 1 TO A1 SET 2
 50 LET F = (1/I) * S
 60 LET P = P+F
 70 LET S = -S
 80 NEXT I
 90 LET P = 4 * P
100 PRINT "VALOR DE PI PELA FORMULA DE LEIBNITZ", P
110 END
```

Resultados:

Para A = 1	A = 2	A = 3
A1 = 2x1 - 1 = 1	A1 = 3x2 - 1= 5	A1 = 3x2 - 1 = 5
S = 1	S = 1	S = 1
I = 1	I = 1 I = 2	I = 1, 3, 5
F = 1	F = 1	F = - (1/3)
P = 1	P = 1 P = 1 - (1/3)	P = 1, - (1/3) + (1/5)
S = - 1	S = -1 S = 1	S = -1, 1 -1
P = 4 x 1 = 4	P = 4 x (1 - 1/3)	P = 4 x (1 - (1/3) + (1/5))
Valor π : 4	: 2.6666664	: 3.466668
Para: A = 60	π = 3.12492706; A = 100	π = 3.1315927

11.6 EXERCÍCIOS DE APLICAÇÃO

1. Escrever um único programa Basic que incorpore as cinco fórmulas de cálculo de π, permitindo a escolha de cada fórmula através de mensagens e de código do tipo:

   DESEJA METODO I – FORMULA CHINESA DO ANO 470A.C.? TECLAR 1

   DESEJA METODO II – FORMULA INDIANA DE ARYABHATA DE 510A.C.? TECLAR 2

   DESEJA METODO III – FORMULA DO MATEMATICO INGLES J. WALLIS? TECLAR 3

   DESEJA METODO IV – FORMULA DO MATEMATICO AUSTRIACO STRASSNITZKY? TECLAR 4

   DESEJA METODO V – FORMULA DE LEIBNITZ? TECLAR 5

   Uma vez optado o código numérico, cada fórmula deve ser fornecida ao usuário juntamente com o resultado.

   (sugestão: usar comando **ON** x **GOTO**. . .)

2. Escrever um programa Basic que forneça a maior precisão possível do valor através de cada uma das cinco fórmulas. Usar precisão simples e depois, se possível, dupla precisão. Efetue uma análise comparativa dos métodos, quanto à precisão e número de operações envolvidas.

Aplicação: Emissão de Extrato de Conta Corrente 12

12.1 O PROBLEMA

Um cliente de um banco possui os dados pessoais de sua conta corrente (nome, número e saldo anterior) e um resumo dos movimentos efetuados durante um certo mês (dez/85).

Dados Pessoais do Correntista	*Variável*
Nome do cliente: José Alves Fonseca	(N$)
Número da conta: 78-3451	(C$)
Saldo anterior: $33457.80	(S)
Movimento do mês de: Dez/85	
Resumo	

Data (D$)	*Operação(P$)*	*Nº documento (X$)*	*Valor (V)*
01/12/85	débito	3478-1	1000,00
10/12/85	crédito	350	5500.00
15/12/85	débito	3479-1	3000,00
27/12/85	débito	3481-1	15000.00
29/12/85	crédito	370	34789.50

12.2 RESULTADO DESEJADO

Queremos ler os dados e emitir um Extrato do Movimento do Mês, do tipo:

NOME:

NUMERO DA CONTA

EXTRATO DO MES DE:

DATA	Nº DOCUMENTO	DEBITO	CREDITO	SALDO
xx/xx/xx	xxxxxxx-x			
		SALDO A TRANSPORTAR		

12.3 PROGRAMA

```
10 READ M$                                   (leitura do mês)
20 READ N$                                   (leitura do nome)
30 READ C$                                   (leitura do número)
40 READ S                                    (leitura do saldo)
50 PRINT
60 PRINT
70 PRINT "NOME:", N$
80 PRINT "NUMERO DA CONTA:"; C$
90 PRINT "EXTRATO DO MES DE: "; M$
100 PRINT "______________________________ "
110 PRINT "DATA     Nº DOCUMENTO     DEBITO   CREDITO   SALDO"
120 PRINT " ____________________________________________ "
130 PRINT "                              SALDO ANTERIOR S", S
140 READ D$, P$, X$, V                       (leitura de um movimento)
150 IF D$ = "999999" THEN 270                (fim do movimento)
160 IF P$ = "DEBITO" THEN 200
```

```
170 IF P$ = "CREDITO" THEN 230
180 PRINT "ERRO DE DEBITO/CREDITO"
190 GO TO 140
200 LET S = S -V
210 PRINT D$,"b"; X$; TAB (5); V, ,S
220 GO TO 140
230 LET S = S + V
240 PRINT D$,"b"; X$, TAB (15); V, S
250 GO TO 140
260 REM FIM DO MOVIMENTO
270 PRINT"                SALDO A TRANSPORTAR$", S
280 PRINT " _________________________"
300 REM DADOS DO CLIENTE
310 DATA "DEZ/85"                        (mês)
320 DATA "JOSE ALVES FONSECA"            (nome do cliente)
330 DATA "78-3451"                       (número da conta)
340 DATA 33457.80                        (saldo anterior)
350 REM MOVIMENTO DO MES
360 DATA "01/12/85", "DEBITO", "3478-1", 1000.00
370 DATA "10/12/85", "CREDITO", "350", 5500.00
380 DATA "15/12/85", "DEBITO", "3479-1", 3000.00
390 DATA "27/12/85", "DEBITO", "3481-1", 15000.00
400 DATA "29/12/85", "CREDITO", "370", 34789.50
410 DATA "999999", "9999", "9999", 99999       (fim do movimento)
500 END
```

Nota: Para outras movimentações da conta, usar mais comandos DATA.

12.4 RESULTADO

```
RUN
NOME: JOSE ALVES FONSECA
NUMERO DA CONTA: 78-3451
EXTRATO DO MES DE: DEZ/85
------------------------------------------------------------------
DATA      Nº DOCUMENTO      DEBITO        CREDITO        SALDO
------------------------------------------------------------------
                            SALDO ANTERIOR           $ 33457.80
01/12/85    3478-1          1000.00                     32457.80
10/12/85    350                            5500.00      37957.80
15/12/85    3479-1          3000.00                     34957.80
27/12/85    3481-1         15000.00                     19957.80
29/12/85    370                           34789.50      54747.30
                       SALDO A TRANSPORTAR $            54747.30
==================================================================
```

12.5 EXERCÍCIOS DE APLICAÇÃO

1. Na aplicação estudada, acertar a coluna dos valores, colocar CR$, ponto para cada mil cruzeiros e vírgula decimal, usando o comando **PRINT USING**.
2. Na prática é necessário processar a conta corrente de um número elevado de clientes.

   Os dados pessoais de cada cliente formam um conjunto denominado:

   ARQUIVO-MESTRE DE CLIENTE, contendo nome, número da conta e saldo de todos os N clientes da agência bancária. O movimento mensal forma um outro arquivo denominado:

   ARQUIVO DE MOVIMENTO MENSAL, que contém, para cada cliente que movimentou a conta, nesse mês, os seguintes dados:
   – número da conta X
   – resumo do movimento dessa conta X :

   01/12/85 DEBITO, Nº DOCUMENTO, VALOR
   03/12/85. . .
   9999999 (indicadora de fim de movimento)

Lembrar que nem todos os clientes do ARQUIVO-MESTRE movimentaram a sua conta.

É necessário formar esses dois arquivos em disquete ou fita cassete, pois não é possível colocar todos os dados dos N clientes nos comandos **DATA.**

Questões:

- Escreva um programa Basic para emissão de extrato de 10 clientes, utilizando somente os comandos **DATA.**
- Transformar o programa para ler os dados de dois arquivos que estão em disquetes.

Programas de Simulação e Jogos de Azar 13

13.1 JOGO DE PAR OU ÍMPAR OU CARA-COROA – COMANDO-FUNÇÃO : RND(X)

Cada um dos dois eventos (CARA ou COROA) tem mesma chance de ocorrer. Usamos a função **Y = RND(X)** que gera automaticamente um valor ao acaso, compreendido entre os valores 0 e 1.

Se Y for menor que o valor 0,5, dizemos que "ocorreu CARA", pois representa 50 % das chances possíveis de Y; caso contrário (Y maior ou igual a 0,5), dizemos que "ocorreu COROA" X.

```
NEW
10 LET X = 0.75           (valor inicial para o gerador RND(X))
20 LET Y = RND(X)         (gera um valor ao acaso entre 0 e 1)
30 IF Y < 0.5 THEN 60
40 PRINT "OCORREU COROA"
50 GO TO 80
60 PRINT "OCORREU CARA"
80 END
```

13.2 SIMULAÇÃO DO DADO DE SEIS FACES

Neste caso, temos seis eventos (face 1, 2, 3, 4, 5 e 6) com as mesmas chances de ocorrência. Então dividimos o intervalo entre 0 e 1 em 6 partes iguais correspondendo, cada subintervalo, à chance de ocorrência de uma face.

```
NEW
10 LET X = 0.75
20 LET Y = RND(X)
30 IF Y< 1/6 THEN 110
40 IF Y< 2/6 THEN 120
50 IF Y< 3/6 THEN 130
60 IF Y< 4/6 THEN 140
70 IF Y< 5/6 THEN 150
80 PRINT "SAIU FACE 6"
90 GO TO 250
110 PRINT "SAIU FACE 1"
115 GO TO 250
120 PRINT "SAIU FACE 2"
125 GO TO 250
130 PRINT "SAIU FACE 3"
135 GO TO 250
140 PRINT "SAIU FACE 4"
145 GO TO 250
150 PRINT "SAIU FACE 5"
250 END
```

Ou, um programa mais econômico seria:

```
NEW
10 LET X = 0.75
20 LET Y = RND(X)
30 FOR I = 1 TO 6
40 IF Y< I/6 THEN 70
50 NEXT I
60 STOP
70 PRINT "SAIU FACE", I
80 END
```

13.3 FORNECIMENTO DE VALORES RANDÔMICOS EQUIPROVÁVEIS PELA FUNÇÃO **RND(X)**

A função **RND(X)** fornece uma seqüência de valores aleatórios ou randômicos equiprováveis (valores ao acaso) entre zero e um, que podem ser utilizados em problemas de simulação e jogos. Para fornecer, por exemplo, 10 valores equiprováveis usamos o programa abaixo.

Programa Basic:

```
10 LET X = 0.35
20 FOR I = 1 TO 10
30 LET Y = RND(X)
40 PRINT "VALOR ALEATORIO", I, Y
50 NEXT I
60 END
```

Resultado:

```
RUN
VALOR ALEATORIO      1      0.857316
VALOR ALEATORIO      2      0.173578
. . .
VALOR ALEATORIO      10     0.975013
```

Nota: Alterando o comando **10 LET X = 0.35**, podemos fornecer valor "de partida ou semente" diferente e ter seqüência diferente de números randômicos. No comando **30 LET Y = RND(X)** em geral não é necessário "renovar" o valor X pelo novo valor gerado Y. Entretanto, em algumas versões essa renovação não é automática e devemos usar:

```
30 LET Y = RND(X)
35 LET X = Y
```

13.4 REPETIÇÃO DOS JOGOS DA MOEDA OU DADOS

Nos exercícios anteriores só podemos jogar um único lance. Para termos lances diferentes devemos mudar o valor inicial (semente) do comando **10 LET X = 0.75.**

Se quisermos jogar lances repetidos podemos usar as seguintes alternativas:

a) Para cinco lances consecutivos:

```
 10 LET X = 0.75              (ou valor qualquer entre 0 e 1)
 15 FOR I = 1 TO 5
 20 LET Y = RND(X)
 30 . . .                     (jogo da moeda ou dado)
250 NEXT I
260 END
```

b) Para decidir se continuamos jogando ou não:

```
 10 LET X = 0.75
 20 LET Y = RND(X)
 30 . . .                   (jogo da moeda ou dado)
 40 . . .

250 PRINT "QUER CONTINUAR? ESCREVA SIM OU NAO"
260 INPUT S$
270 IF S$ = "SIM" THEN 20
280 END
```

Observação: Para o mesmo valor inicial (somente) X teremos a mesma seqüência de valores **RND(X)** gerados ao acaso. Alterando X teremos seqüências diferentes.

13.5 SIMULAÇÃO DE UM DADO VICIADO

A maior vantagem da simulação é que, podemos alterar, à vontade o programa e introduzir modificações ou vícios difíceis de se conseguir no caso real.

Por exemplo, queremos que o dado simulado seja VICIADO de modo que os valores pares tenham duas vezes mais chances de sair do que os ímpares.

Então temos:	Face 1: Peso 1 ou Chance 1/9 (no total de 100%)	
	Face 2: Peso 2 ou Chance 2/9	
	Face 3: Peso 1 ou Chance 1/9	
	Face 4: Peso 2 ou Chance 2/9	
	Face 5: Peso 1 ou Chance 1/9	
	Face 6: Peso 2 ou Chance 2/9	
	Total 9	1 ou 100%

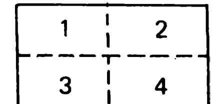
Basta introduzir no programa anterior, intervalos com essas proporções, isto é, Y gerado será comparado com os valores 1/9, 3/9, 4/9, 6/9, 7/9 em vez de 1/6, 2/6, 3/6, 4/6, 5/6.

13.6 SIMULAÇÃO DE UMA BATALHA INTERPLANETÁRIA

Descrição do Problema:

TELA DO RADAR

A tela do RADAR da Base de Defesa Terrestre está dividida em quatro quadrados numerados 1, 2, 3 e 4.



TELA DO RADAR DA BASE TERRESTRE

"UFO" INVASOR

Uma espaçonave invasora (UFO invasor) pode surgir em qualquer quadrado da tela, e seus mísseis:

- têm 100% de chance de DESTRUIR A BASE (ao menos que seja ABATIDA) se o UFO estiver no quadrado 3 ou 4 e
- têm 50% de chance de DESTRUIR A BASE, se o UFO estiver no quadrado 1 ou 2 (outros 50% de chance é de que o UFO passou sem atingir a BASE).

DEFESA DA BASE

A Base está equipada com:

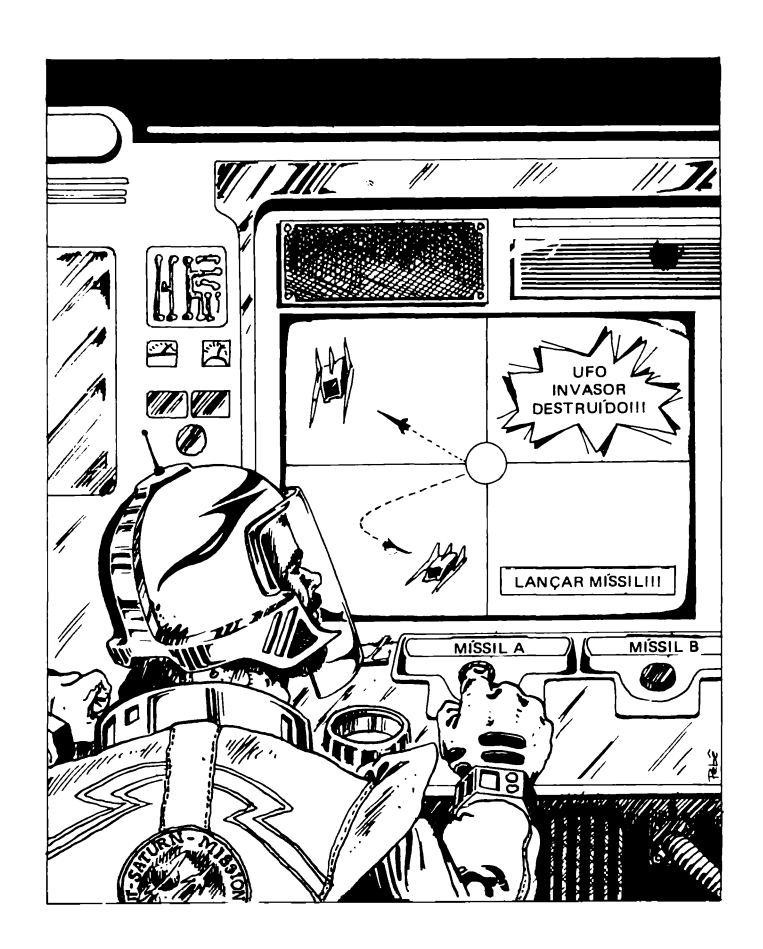
- Três mísseis do tipo A cuja chance de acerto é de 70% e
- Dois mísseis do tipo B cuja chance de acerto é de 90%

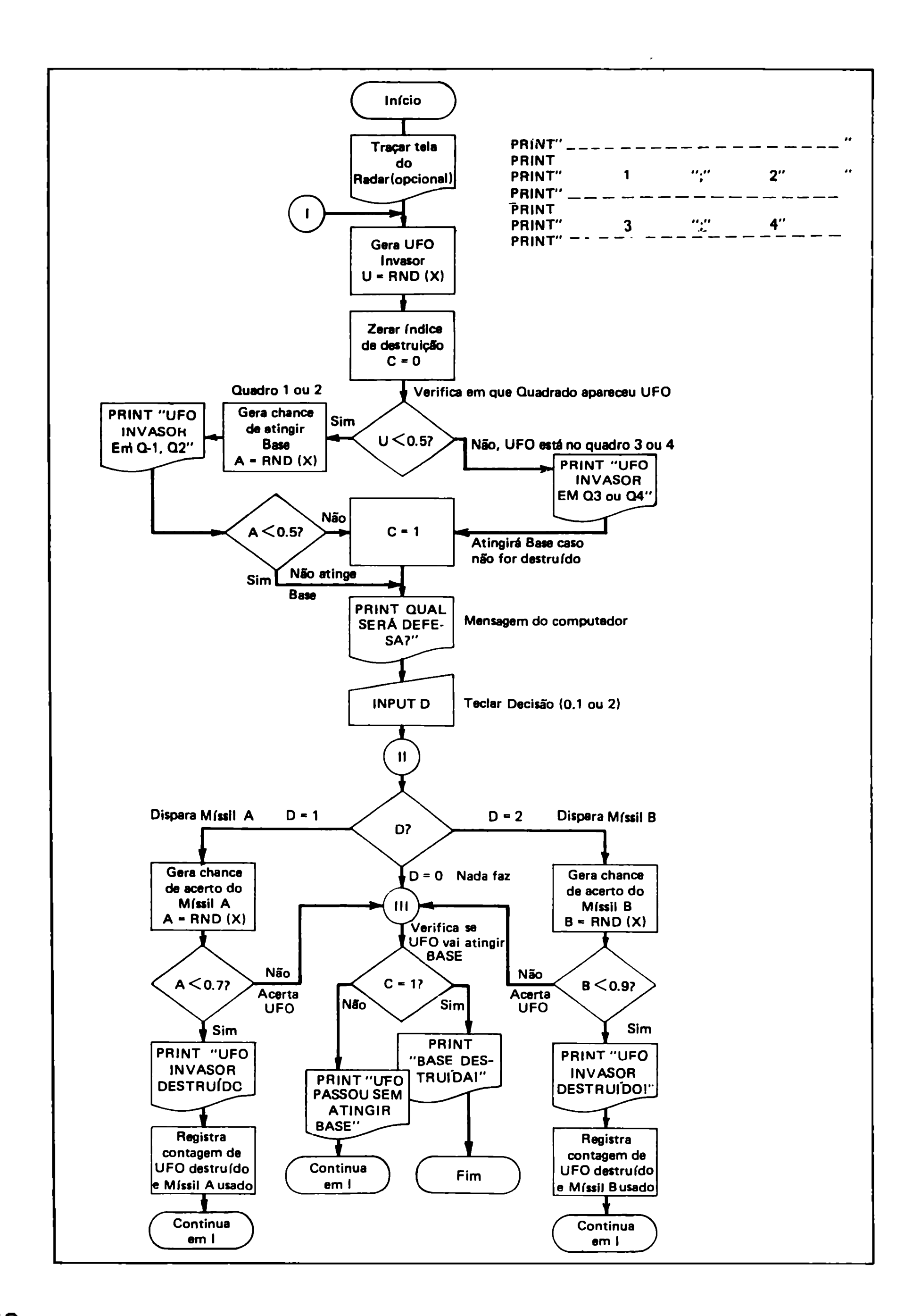
DECISÃO A SER TOMADA

A cada UFO invasor que aparecer na tela, deve-se tomar uma das seguintes decisões: (até que a BASE ou o UFO seja destruído)

- Disparar um Míssil do tipo A (teclar valor 1)
- Disparar um Míssil do tipo B (teclar valor 2)
- Somente observar se o UFO desaparece (teclar valor 0).

Observação: Na tela de TV do computador usado, poder-se-ia traçar graficamente a trajetória reta (com certo ângulo de inclinação gerado ao acaso) ou curva dos mísseis lançados. A destruição do UFO invasor ocorrerá se a trajetória passar pelo ponto de coordenada do UFO. (Fig. 13.1)





Início
Traçar tela do Radar(opcional)
PRINT" _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ "
PRINT
PRINT" 1 ";" 2" "
PRINT" _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
PRINT
PRINT" 3 ";" 4"
PRINT" _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
I
Gera UFO Invasor U = RND (X)
Zerar índice de destruição C = 0
Verifica em que Quadrado apareceu UFO
Quadro 1 ou 2
U <0.5?
Sim
Gera chance de atingir Base A = RND (X)
PRINT "UFO INVASOR Em Q-1, Q2"
Não, UFO está no quadro 3 ou 4
PRINT "UFO INVASOR EM Q3 ou Q4"
Atingirá Base caso não for destruído
A <0.5?
Não
C = 1
Sim
Não atinge Base
PRINT QUAL SERÁ DEFE- SA?"
Mensagem do computador
INPUT D
Teclar Decisão (0.1 ou 2)
II
D?
Dispara Míssil A
D = 1
D = 2
Dispara Míssil B
D = 0 Nada faz
Gera chance de acerto do Míssil A A = RND (X)
Gera chance de acerto do Míssil B B = RND (X)
III
Verifica se UFO vai atingir BASE
A <0.7?
Não
Acerta UFO
C = 1?
Não
Sim
Não
Acerta UFO
B <0.9?
Sim
Sim
PRINT "UFO INVASOR DESTRUÍDC
PRINT "BASE DES- TRUÍDA!"
PRINT "UFO PASSOU SEM ATINGIR BASE"
PRINT "UFO INVASOR DESTRUÍDO!"
Registra contagem de UFO destruído e Míssil A usado
Registra contagem de UFO destruído e Míssil B usado
Continua em I
Fim
Continua em I
Continua em I

Programa Basic:

```
0005 REM TRACAR TELA DO RADAR
0010 PRINT "_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
0020 PRINT "          1        I           2  "
0030 PRINT "_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
0040 PRINT "          3        I           4  "
0050 PRINT "_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
0053 LET A1 = 3
0054 LET B1 = 2
0055 LET X = .754
0060 LET U = RN D(X)
0061 PRINT "U", U
0080 LET C  = 0
0085 REM VERIFICA EM QUE QUADRO APARECEU UFO
0090 IF U< . 5 THEN 120
0094 PRINT "UFO INVASOR EM QUADRO 1 - 2"
0095 LET A = RND(X)
0097 PRINT "A", A          (opcional)
0098 IF A< 0.5 THEN 150
0100 GOTO 160
0120 PRINT "UFO INVASOR EM QUADRO 3 - 4"
0150 LET C = 1
0160 PRINT "QUAL SERA DEFESA?"
0162 PRINT "NADA FAZ = TECLAR ZERO"
0163 PRINT "DISPARA MISSIL A = TECLAR 1"
0164 PRINT "DISPARAR MISSIL B = TECLAR 2"
0165 INPUT D
```

```
0170 IF D = O THEN 200
0175 IF D > = 2 THEN 300
0180 LET A1 = A1 - 1
0181 LET A = RND(X)
0182 PRINT "A", A                    (opcional)
0184 IF A > .7 THEN 200
0185 PRINT "UFO INVASOR DESTRUIDO!!"
0190 PRINT "RESTAM", A1, B1;" MISSEIS A E B"
0195 GOTO 60
0199 REM NADA FAZ OU MISSIL NAO ATINGE UFO INVASOR
0200 IF C = 1 THEN 230
0210 PRINT "UFO PASSOU SEM ATINGIR BASE"
0212 PRINT "RESTAM "; A1, B1; "MISSEIS A E B"
0215 GOTO 60
0230 PRINT "BASE DESTRUIDA! ! !"
0241 GOTO 500
0299 REM MISSIL B
0300 LET B = RND(X)
0301 PRINT "B", B                    (opcional)
0305 LET B1 = B1 - 1
0320 IF B > .9 THEN 200
0325 GOTO 185
0500 END
```

Resultados:

```
#RUN

— — — — — — — — — — — — —
  1     I     2
— — — — — — — — — — — — —
  3     I     4
— — — — — — — — — — — — —
```

```
U                              0.36489123     (valor aleatório
UFO INVASOR EM QUADRO 3-4                     de verificação)
QUAL SERA DEFESA?
NADA FAZ = TECLAR ZERO
DISPARA MISSIL A = TECLAR 1
DISPARAR MISSIL B = TECLAR 2
? 2                                           (decisão tomada)
B                              0.07006553
UFO INVASOR DESTRUIDO! !
```

```
RESTAM 2                       1  MISSEIS A E B
U                              0.8572666
UFO INVASOR EM QUADRO 1-2
A                              0.10350774
QUAL SERA DEFESA?
NADA FAZ = TECLAR ZERO
DISPARA MISSIL A = TECLAR 1
DISPARAR MISSIL B = TECLAR 2
? 1                                           (decisão tomada)
A                              0.68622667
UFO INVASOR DESTRUIDO! !
```

```
RESTAM 1                    1  MISSEIS A E B
U                           0.7144645
UFO INVASOR EM QUADRO 1-2
A                           0.93888811
QUAL SERA DEFESA?
NADA FAZ = TECLAR ZERO
DISPARA MISSIL A = TECLAR 1
DISPARAR MISSIL B = TECLAR 2
? 1                                            (decisão tomada)
A                           0.89887604
UFO PASSOU SEM ATINGIR BASE
```

```
U                           0.57187251
UFO INVASOR EM QUADRO 1-2
A                           0.76402883
QUAL SERA DEFESA?
NADA FAZ = TECLAR ZERO
DISPARA MISSIL A = TECLAR 1
DISPARAR MISSIL B = TECLAR 2
? 1                                            (decisão tomada)
A                           0.33286963
UFO INVASOR DESTRUIDO! !
```

```
RESTAM – 0                  1  MISSEIS A E B
U                           0.74896865
UFO INVASOR EM QUADRO 1-2
A                           0.21277704
QUAL SERA DEFESA?
NADA FAZ = TECLAR ZERO
DISPARA MISSIL A = TECLAR 1
DISPARAR MISSIL B = TECLAR 2
```

```
? 2                                              (decisão tomada)
B                        0.74648813
UFO INVASOR DESTRUIDO! !
```

```
RESTAM – 0               0. MISSEIS A E B
                         0.22900508
UFO INVASOR EM QUADRO 3-4
QUAL SERA DEFESA?
NADA FAZ = TECLAR ZERO
DISPARA MISSIL A = TECLAR 1
DISPARAR MISSIL B = TECLAR 2
? 0                                              (decisão tomada)
BASE DESTRUIDA! ! !
```

13.7 EXERCÍCIOS DE APLICAÇÃO

1. Escrever um programa Basic para simular o lançamento de uma moeda dez vezes repetidas. Contar o número de "caras" ocorridas.

2. Escrever um programa Basic para simular o lançamento simultâneo de dois dados.

   Repetir a simulação por dez vezes e calcular:

   – o número de faces pares
   – o número de faces com valor menor ou igual a 3
   – a média da soma dos valores dos dois dados.

3. Simular, em Basic, o jogo do Black-Jack ou Sete e meio.

4. Escrever um programa que simule o jogo Tic-Tac-Toc ou jogo-da-velha. O computador escolhe qualquer posição vaga ao acaso.

5. Modificar o problema de SIMULAÇÃO DA BATALHA INTERPLANETÁRIA para JOGO DE COBRANÇA DE PENALIDADES MÁXIMAS EM FUTEBOL, substituindo os mísseis A e B por CHUTE DE PÉ DIREITO e CHUTE DE PÉ ESQUERDO, respectivamente, e os Quadrados 1, 2, 3 e 4 da tela do Radar por DIREÇÕES 1, 2, 3 e 4 dos chutes nos quatro quadrados em que está dividido o GOL. Os chutes podem ir para o gol ou para fora com as mesmas chances dadas para Acerto ou Erro do Míssil. Nas Direções 1 e 2 o Goleiro tem 50% de chance de defender a penalidade. Escrever o Fluxograma e o programa BASIC que efetue a cobrança de três penalidades máximas e registre no placar.

Aplicação: Computer-Assisted-Learning (CAL): o Ensino Através do Computador



14.1 O ENSINO ATRAVÉS DE MICROCOMPUTADOR E BASIC

O computador tem-se mostrado um instrumento útil e eficiente no ensino, tanto no currículo oficial como no treinamento profissional nas empresas. (Fig. 14.1)

Algumas das vantagens proporcionadas pelo ensino através do computador são:

- participação ativa e repetitiva do aluno nas lições fornecidas pelo computador, com treinamento programado e progressivo;
- possibilidade de preparar textos e exercícios de qualquer tipo e finalidade;
- baixo custo proporcionado pelos microcomputadores e pela linguagem Basic.

Alguns campos de aplicação do **CAL** (ensino através do computador) são:

- laboratório de treinamento de línguas, ciências e matemática;
- treinamento profissional para um novo cargo, operação de novo equipamento etc.;
- treinamento de excepcionais e pessoas com dificuldades físicas ou mentais.

14.2 PROGRAMA EDUCACIONAL PARA CRIANÇAS: EXERCÍCIOS DE ADIÇÃO

Programa para ensinar operação de adição de um algarismo.

```
0001 PRINT "ENSINO POR COMPUTADOR – EXERCÍCIO DE SOMA
     PARA CRIANÇAS"
0005 LET K = 0.781
0010 PRINT
0011 LET Z = RND(K)
0020 LET X = INT (10 * Z + 1)
0021 LET Z = RND(X)
0030 LET Y = INT (10  * Z + 1)
0040 PRINT X; "+", Y, "="
0050 INPUT R
0070 IF R = X + Y THEN 100
0080 PRINT "NAO, ",X;"+" ;Y," = ",X + Y
0090 GOTO 10
0100 PRINT "CORRETO! TENTE MAIS ESTA"
0110 GOTO 10
0120 END
```

```
# RUN
ENSINO POR COMPUTADOR – EXERCICIO DE SOMA PARA
     CRIANÇAS
6 + 7 =
? 13
CORRETO! TENTE MAIS ESTA
4 + 10 =
? 15
NAO, 4 + 10 = 14
10 + 8 =
? 18
CORRETO! TENTE MAIS ESTA
3 + 5 =
? 8
CORRETO! TENTE MAIS ESTA
4 + 6 =
? 11
NAO, 4 + 6 = 10
6 + 2 =
?
```

Nota: Adaptado de: *Mathematics Teaching* – Travers; K, e outros. Harper & Row – 1977 – USA.

14.3 PROGRAMA EDUCACIONAL PARA CRIANÇAS: EXERCÍCIOS DE MULTIPLICAÇÃO DE DOIS NÚMEROS DE UM ALGARISMO CADA UM

```
 10 PRINT
 15 LET K = 0.791
 20 LET X = INT (10 * RND(K) + )
 30 LET Y = INT (10 * RND(K) + 1)
 40 PRINT X; "X" ;Y; "="
 50 PRINT R
 60 IF R = X + Y THEN 90
 70 PRINT "NAO" ;X; "X" ;Y; "=" ;X * Y
 80 GO TO 20
 90 PRINT "CORRETO! TENTE MAIS ESTA"
100 GOTO 20
110 END
```

```
RUN
6  3 = ? 18
CORRETO! TENTE MAIS ESTA
7  9 = ? 64
NAO, 7  9 = 63
8  2 = ?
```

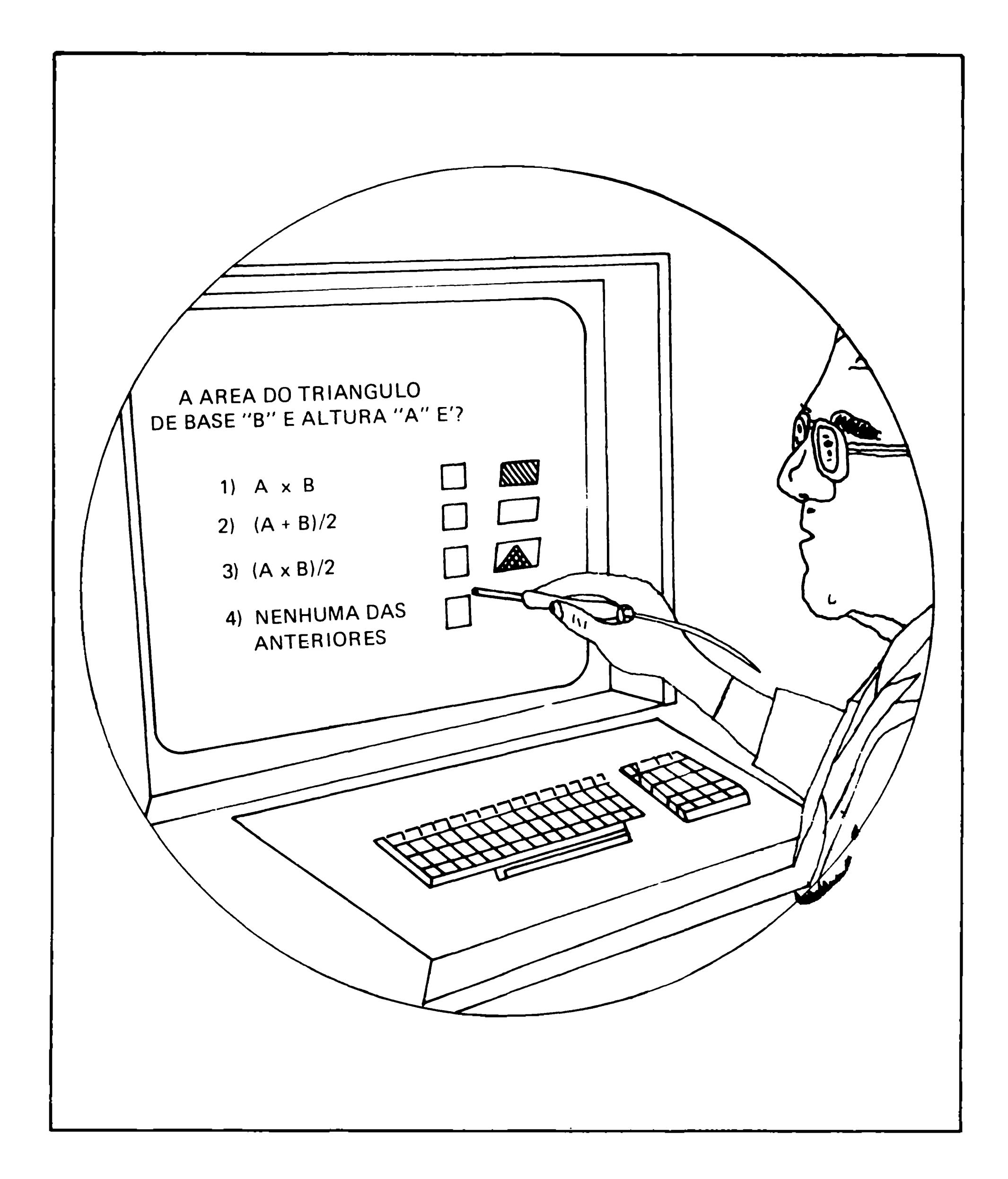


Figura 14.1. *Programa BASIC para o ensino CAI*

14.4 PROGRAMA EDUCACIONAL: EXERCÍCIO DE FATORAÇÃO

Fórmula: $(A^2 - B^2) = (A + B)(A - B)$

```
 10 PRINT "FATORACAO DA DIFERENCA DE DOIS QUADRADOS"
 20 PRINT" (A ↑ 2 - B ↑ 2 ) = (A + B) (A - B)"
 30 PRINT
 40 PRINT "EXEMPLO: (A ↑ 2 - 36) = (A +6) (A – 6)"
 45 LET K = 0.378
 50 PRINT
 60 LET A = INT (11 * RND(K) + 1)
 70 LET C = A * A
 80 PRINT "A ↑ 2 - " ;C; "= (A + B) (A – B)"
 90 PRINT "QUANTO VALE B"
100 INPUT B
110 IF A = B THEN 150
120 PRINT "NAO, PARA O SEU VALOR B, B ↑ 2 = "; B * B
130 PRINT "TENTE NOVAMENTE."
140 GO TO 80
150 PRINT "CORRETO! (A ↑ 2 -" ;C;") = (A + ";B;") (A – ";B;")"
160 GO TO 50
170 END
```

Observação: Verifique no manual do seu microcomputador o uso correto de RND(K).

```
RUN
FATORACAO DA DIFERENCA DE DOIS QUADRADOS
(A ↑ 2 - B ↑ 2) = (A + B) (A - B)
EXEMPLO: (A ↑ 2 - 36) = (A - 6) (A + 6)
A ↑ 2 - 16 = (A + B) (A - B)
QUANTO VALE B?4
CORRETO! (A ↑ 2 - 16 ) = (A + 4) (A - 4)
A ↑ 2 - 49 = (A + B) (A - B)
QUANTO VALE B?8
NAO, PARA O SEU VALOR B, B ↑ 2 = 64
TENTE NOVAMENTE
A ↑ 2 - 49 = (A + B) (A - B)
QUANTO VALE B?
```

14.5 EXERCÍCIOS DE APLICAÇÃO

1. Escreva os programas educacionais para crianças para exercícios de subtração e divisão. Testar a eficiência didática, submetendo-os ao uso de várias crianças.

2. Os programas educacionais para exercícios de adição e multiplicação fornecem parcelas ou fatores de 1 a 9. Modifique os programas para fornecer valores entre 1 e 100.

   (Sugestão: gerador de valores passa a ser: **X = INT (100 * RND(K) + 1))**

3. Discutir a implicação didática ocorrida nos programas educacionais devido à modificação introduzida pelo exercício anterior.

   Ficou difícil demais para uma criança?

Introduzir melhoramentos didáticos no programa, por exemplo:

DISPOSIÇÃO VERTICAL DAS OPERAÇÕES:	OU	DESDOBRAR EM DUAS OPERAÇÕES DE UM ALGARISMO CADA:
34 + 87 ——— ? 121		4 + 7 = ? 11 CORRETO! RESULTADO IGUAL A 1 COM VAI UM 3 + 8 + (VAI UM) = 12 CORRETO! RESULTADO IGUAL A 121

4. Escrever e testar programas educacionais para exercícios de:

   – **(A + B) ↑ 2 = A ↑ 2 + 2A. B + B ↑ 2**

   – Dado um ângulo T, fornecer o seu complementar e suplementar.

Algoritmo de Aproximações Sucessivas: Raiz Quadrada, Raiz Cúbica e Raiz de Equações 15

15.1 RAIZ QUADRADA DE UM NÚMERO: ALGORITMO DE APROXIMAÇÃO SUCESSIVAS

Um algoritmo simples, para obter a raiz quadrada de um número qualquer, através de fórmulas de aproximação sucessiva do tipo iterativo, é dado pelo seguinte programa:

Programa Basic:

```
05 PRINT "RAIZ QUADRADA DO VALOR N POR APROXIMACOES
SUCESSIVAS"
06 PRINT "FORNECER VALOR N"
10 INPUT N
20 LET K = 2
30 LET Q = N/K
35 PRINT "VALOR APROXIMADO DA RAIZ", Q
40 IF ABS (K - Q) < 0.0001 THEN 70
50 LET K = (K + Q)/2
60 GO TO 30
70 PRINT "RAIZ QUADRADA DE" ,N, "IGUAL A", Q
80 END
```

Resultado: (Teste de verificação)

Para: **N = 9**

K = 2

Q = 9/2 = 4.5

/2 – 4.5/ = 2.5

K = (2 + 4.5)/2 = 3.25

VALOR APROXIMADO DA RAIZ: 4.5

VALOR APROXIMADO DA RAIZ: 2.76923

VALOR APROXIMADO DA RAIZ: 2.9904157

VALOR APROXIMADO DA RAIZ: 2.999846

RAIZ QUADRADA DE 9 IGUAL A 2.9999

Para: **N = 2**

VALOR APROXIMADO DA RAIZ:	**1**
	1.33334
	1.4117614
	1.4142114
RAIZ QUADRADA DE 2 IGUAL A	**1.4142114**

Observação: Maior precisão do resultado é possível alterando-se o comando:

40 IF ABS (K - Q) < 0.0001 THEN 70

15.2. RAÍZES DE EQUAÇÕES

Em problemas de engenharia e ciência é freqüente a determinação das raízes de alguma equação. Raízes ou zeros de uma equação são os valores de x tais que f(x) = 0.

As equações podem ser polinominais e transcendentais:

$x^2 + 3x + 2 = 0$

$5x^3 + 2x = 0$

$x - \text{sen}^2 x = 0$

Existem algumas regras que ajudam o processo de achar o intervalo onde estão as raízes de equações polinominais, o que é mais difícil para equações transcendentais do tipo $x - sen^2 x = 0$.

15.3 MÉTODOS PARA A OBTENÇÃO DE UMA RAIZ

Uma vez determinado um intervalo (a, b) onde se sabe que existe uma raiz, aplica-se um dos métodos de determinação da raiz.

15.3.1 Método de Newton-Raphson

É o método utilizado devido a sua rápida convergência.

Dado um ponto x_1 pertencente ao intervalo (a,b) onde existe uma raiz da equação $f(x) = 0$, a raiz será obtida pela fórmula iterativa:

$$x_{n+1} = x_n - (f(x_n)/f'(x_n))$$

onde $f'(x_n)$ é a derivada de $f(x)$ no ponto x_n.

O método não converge se $f'(x_1) = 0$. O processo iterativo termina se a diferença $|x_{n+1} - x_n|$ for suficientemente pequena.

Exemplos:

1. Achar a raiz da função:

$$f(x) = x^3 - 23x^2 + 62x - 40 = 0$$

$$\text{A derivada é: } f'(x) = 3x^2 - 46x + 62$$

Partindo de $x_1 = 21$ temos sucessivamente:

x	f(x)	f'(x)	f(x)/f'(x)	x–f(x)/f'(x)
21,0	180	419	0,91	20,1
20,1	32,2	349	0,09	20,0
20,0	0			

Portanto, uma das raízes é $x_i = 20$.

2. Achar uma raiz de:

$$f(x) = x^2 - \text{sen}\ x = 0$$

A derivada é: $f'(x) = 2x - \cos x$.

x	f(x)	f'(x)	f(x)/f'(x)
1,0	0,159	1,460	0,109
0,891	0,017	1,154	0,014
0,877	0,0		

15.4 RAIZ QUADRADA E RAIZ CÚBICA PELO MÉTODO DE NEWTON-RAPHSON

O método de Newton-Raphson também serve para achar a raiz k-ésima de um valor C dado através da relação $x^k = C$ ou $x = \sqrt[k]{C}$.

Por exemplo, para calcular a raiz quadrada do valor dois, temos:

$C = 2$, $k = 2$ e $x^2 = 2$. Então $f(x) = x^2 - 2$ e $f'(x) = 2x$.

Partindo de $x_1 = 1$ temos:

x	$f(x) = x^2 - 2$	$f'(x) = 2x$	f(x)/f'(x)
1	-1	2	-1/2
1.5	0.25	3	0.0833
1.4167	...	...	...

Para calcular a raiz cúbica de um valor C dado, usamos a função:

$$f(x) = x^3 - C$$

15.5 PROGRAMA BASIC PARA ENCONTRAR RAÍZES DE UM POLINÔMIO ATÉ 5º GRAU

Vamos fazer um programa que ache as raízes da equação dada abaixo:

$$F(x) = x^4 - 9x^3 - 2x^2 + 120x - 130$$

Os intervalos onde se localizam as quatro raízes podem ser determinados através do gráfico que aparece na Figura 15.1

Figura 15.1. *Raízes de um polinômio de quarto grau.*

Programa Basic:

```
0100 PRINT "RAIZES DA EQUACAO"
0200 PRINT "ATE QUINTO GRAU"
0300 PRINT "QUAL E A PRECISAO DESEJADA E QUANTAS
ITERACOES"
0400 INPUT P, J
0450 I = 0
0500 PRINT "ENTRE COM OS COEFICIENTES A, B, C, D, E, F"
0600 INPUT A, B, C, D, E, F
0700 PRINT "DE-ME UM PONTO DE PARTIDA"
0800 GOTO 1000
0900 PRINT "DE-ME OUTRO PONTO DE PARTIDA"
1000 INPUT X
1050 I = 0
1100 X5 = X ↑ 5
1200 X4 = X ↑ 4
1300 X3 = X ↑ 3
1400 X2 = X * X
1500 I = I + 1
1600 G =A * X5 + B * X4 + C * X3 + D * X2 + E * X + F
1700 G1 = ABS(G)
1800 IF I > J THEN GO TO  900
1900 IF G1 < P THEN GO TO 2300
2000 D1 = 5 * A * X4 + 4 * B * X3 + 3 * C * X2 + 2 * D * X + E
2100 LET X = X - G/D1
2200 GOTO 1100
2300 PRINT "X =", X
2400 PRINT "ESCREVA 1 SE QUISER OUTRA RAIZ OU 0 SE NAO"
2500 INPUT R
2600 IF R = 1 THEN GO TO 900
2700 IF R = 0 THEN GO TO 2900
2800 PRINT "1 OU 0 ?"
2900 END
READY
```

```
# RUN
RAIZES DA EQUACAO
ATE QUINTO GRAU
QUAL E A PRECISAO DESEJADA E QUANTAS ITERACOES
? 0.0001, 10
ENTRE COM OS COEFICIENTES A, B, C, D, E, F
? 0, 1, -9, -2, 120, - 130
DE-ME UM PONTO DE PARTIDA
? 1
X =          1.22858939                  (1ª raiz)
ESCREVA 1 SE QUISER OUTRA RAIZ OU 0 SE NAO
? 1
DE-ME OUTRO PONTO DE PARTIDA
? 3
X =          3.97206884                  (2ª raiz)
ESCREVA 1 SE QUISER OUTRA RAIZ OU 0 SE NAO
? 1
DE-ME OUTRO PONTO DE PARTIDA
? 6
X =         -3.60013528                  (3ª raiz)
ESCREVA 1 SE QUISER OUTRA RAIZ OU 0 SE NAO
? 1
DE-ME OUTRO PONTO DE PARTIDA
? 7
X =          7.39947745                  (4ª raiz)
ESCREVA 1 SE QUISER OUTRA RAIZ OU 0 SE NAO
? 0
STOP 2900
READY
```

15.6. EXERCÍCIOS DE APLICAÇÃO

1. Escrever e testar um programa Basic para o exemplo 1 da seção 15.3.1, usado no método de Newton-Raphson.

2. Idem, exemplo 2.

3. Idem, para obter a raiz quadrada de um valor C dado, pelo método de Newton-Raphson.

4. Idem, para obter a raiz cúbica de um valor C dado, pelo método de Newton-Raphson.

Gráficos por Computador Usando Basic 16

16.1 GRÁFICO DE UMA FIGURA TRAÇADA PONTO A PONTO

> **O primeiro passo para traçar um gráfico consiste em colocar a figura em um papel quadriculado ou milimetrado e traçar caractere por caractere o perfil da figura, usando sucessão de comandos PRINT " . . . ."**

No exemplo seguinte temos a imagem de um CAVALO MARINHO.

Programa

```
10 REM GRAFICO POR TABULACAO DIRETA
20 PRINT
30 PRINT TAB (10)."            *"
40 PRINT TAB (10),"           **"
50 PRINT TAB (10);"          ********"
60 PRINT TAB (10);"        ***********"
80 PRINT TAB (10);"      ***************"
120 PRINT TAB (10);"  **********************"
125 PRINT TAB (10);"   ****************************"
130 PRINT TAB (10);"  ****************************"
140 PRINT TAB (10);"**************************"
160 PRINT TAB (10);"  ****************"
180 PRINT TAB (10);"*******************"
200 PRINT TAB (10);"  *******************"
220 PRINT TAB (10);" *******************"
240 PRINT TAB (10);"   *********************"
250 PRINT TAB (10);"  *******************"
260 PRINT TAB (10);"    ******************"
280 PRINT TAB (10);"     ********************"
290 PRINT TAB (10);"     ******************"
300 PRINT TAB (10);"      ******************"
320 PRINT TAB (10);"     *******************"
340 PRINT TAB (10);"     ****************"
350 PRINT TAB (10);"     **************"
360 PRINT TAB (10);"    *********"
380 PRINT TAB (10);"    *******"
420 PRINT TAB (10);"   *******"
440 PRINT TAB (10);"   ******"
460 PRINT TAB (10);"    ******"
470 PRINT TAB (10);"     ******"
480 PRINT TAB (10);"      *******************"
490 PRINT TAB (10);"       *************"
500 PRINT TAB (10);"                   **"
510 PRINT TAB (10);"                     *"
600 END
```

16.2 CODIFICAÇÃO DA FIGURA EM CÓDIGOS NUMÉRICOS

Uma vez efetuado o levantamento ponto a ponto em papel quadriculado, a figura pode ser substituída por uma representação ou codificação numérica das posições dos pontos ou caracteres. Assim eliminamos a sucessão de comandos **PRINT**.

No exemplo seguinte, cada linha da figura da seção 16.1 foi substituída por dois valores numéricos:

P: posição até o primeiro caractere " * " e

N: quantidade de caracteres " * " que compõe a linha.

Os comandos **DATA** contêm a sucessão de valores P e N dos comandos **PRINT**.

Sendo assim:

P = 23 e N = 1, representam 23 posições em branco e 1 caractere " * ";

P = 22 e N = 2, representam 22 posições em branco e 2 caracteres " * "

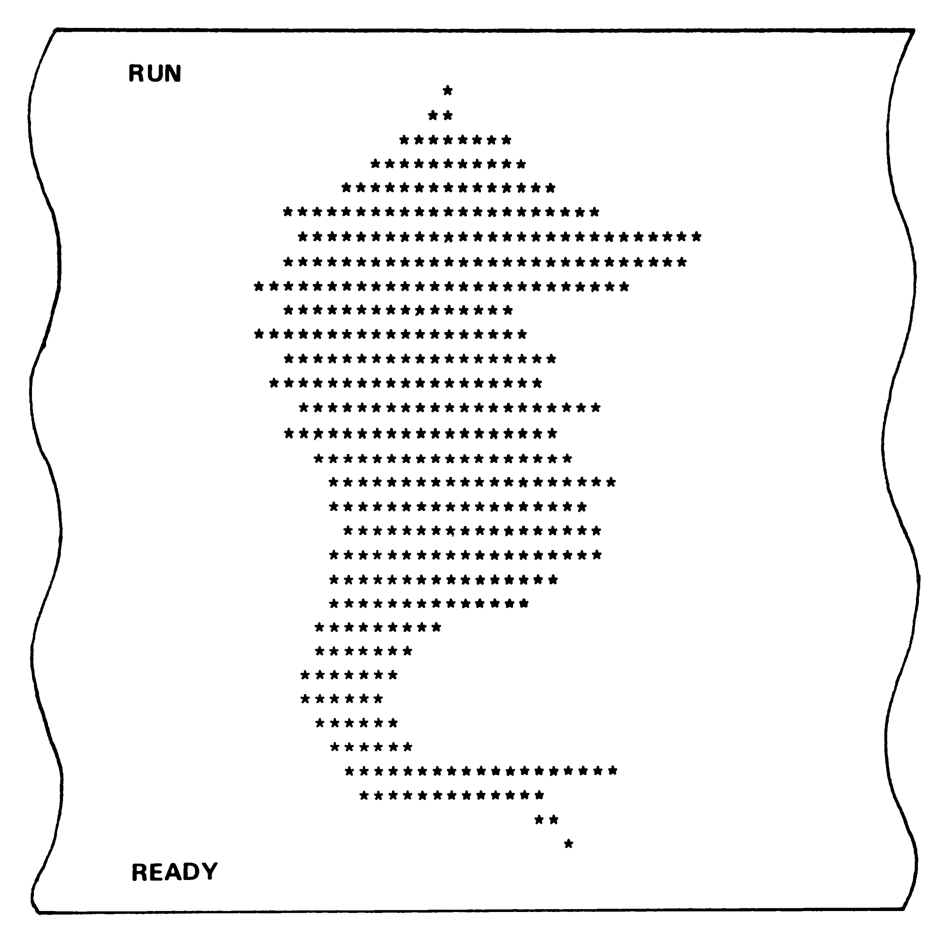
O resultado obtido é o mesmo da seção 16.1.

16.2.1 Exercício

```
30 REM P = POSICAO ATE PRIMEIRO SINAL
40 REM N = NUMERO DE SINAIS
50 READ P, N
60 IF P = 0 THEN IF N = 0 THEN 600
70 PRINT TAB (P);
80 FOR J = 1 TO N
90 PRINT " *";
100 NEXT J
110 PRINT "b̸"
120 GOTO 50
400 DATA 23, 1, 22, 2, 21, 8, 19, 11, 17, 15, 13, 22, 14, 28, 13, 28
410 DATA 11, 26, 13, 16, 11, 19, 13, 19, 12, 19, 14, 21, 13, 19, 15, 18
440 DATA 16, 20, 16, 18, 17, 18, 16, 19, 16, 16, 16, 14, 15, 9, 15, 7
460 DATA 14, 7, 14, 6, 15, 6, 16, 6, 17, 19, 18, 13, 30, 2, 32, 1, 0, 0, 0
600 END
```

Nota: "b̸": espaço em branco.

Respostas dos programas dos itens 16.1 e 16.2



Observação: O trabalho de levantar ponto a ponto uma figura pode ser simplificado em sistemas mais sofisticados, através do uso de equipamento especial, denominado DIGITADOR DE FIGURAS, que transforma o contorno da figura em coordenadas cartesianas.

16.3 TRANSFORMAÇÃO DA FIGURA: DESLOCAMENTO À DIREITA OU À ESQUERDA

A vantagem da representação da figura por uma tabela de valores numéricos reside na facilidade de efetuar transformações, uma vez que estamos manipulando apenas valores numéricos, P e N.

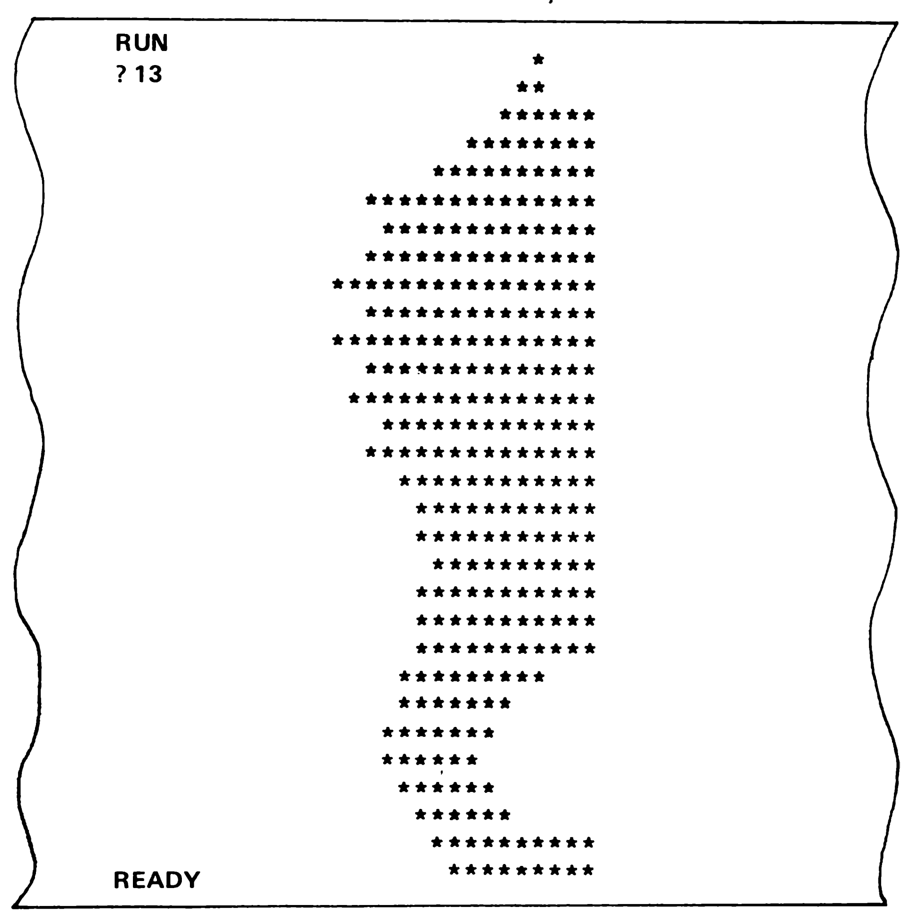
No exemplo, temos um programa que permite deslocamento à direita até um limite máximo de 40 posições. Esse limite foi necessário, pois em BASIC a impressão de mais de 40 caracteres seguidos em uma linha torna o controle da impressão difícil (a linha muda automaticamente) a menos que alguma função especial de controle seja usada.

Programa:

```
 20 REM TRANSLACAO PARA DIREITA – K POSICOES
 30 REM P = POSICAO ATE PRIMEIRO SINAL
 40 REM N = NUMERO DE SINAIS
 45 INPUT K
 50 READ P, N
 60 IF P = 0 THEN IF N = 0 THEN 600
 61 REM TESTE DO LIMITE DE TRANSLACAO (MAX 40 POSICOES)
 62 P = P + K
 63 IF P > 40 THEN 66
 64 IF P + N > 40 THEN 68
 65 GO TO 70
 66 PRINT
 67 GO TO 50
 68 N = 40 - P
 70 PRINT TAB (P);
 80 FOR J = I TO N
 90 PRINT " *",
100 NEXT J
110 PRINT "ƀ"
120 GO TO 50
400 DATA 23, 1, 22, 2, 21, 8, 19, 11, 17, 15, 13, 22, 14, 28, 13, 28
410 DATA 11, 26, 13, 16, 11, 19, 13, 19, 12, 19, 14, 21, 13, 19, 15, 18
440 DATA 16, 20, 16, 18, 17, 18, 16, 19, 16, 16, 16, 14, 15, 9, 15, 7
460 DATA 14, 7, 14, 6, 15, 6, 16, 6, 17, 19, 18, 13, 30, 2, 32, 1, 0, 0, 0
600 END
```

Resultado:

DESLOCAMENTO À DIREITA EM K = 13 POSIÇÕES
FOI FIXADO UM LIMITE À DIREITA NA POSIÇÃO 40.



16.4 DEFORMAÇÃO GRADUAL DA FIGURA: DESLOCAMENTO E ROTAÇÃO

Neste exemplo, é possível introduzir dois tipos de movimentos na figura original.

a) deslocamento à direita ou à esquerda conforme o valor K1 seja positivo ou negativo;
b) rotação gradual em torno do ponto superior da figura de acordo com o valor dado a K2.

Programa:

```
20 REM DEFORMACAO GRADUAL (TRANSLACAO E/OU ROTACAO)
22 REM EM TORNO DA POSICAO SUPERIOR DA FIGURA
23 REM K1 = INCREMENTO DA TRANSLACAO
24 REM K2 = INCREMENTO DA ROTACAO
30 INPUT K1, K2
32 K = K1
33 K3 = 0
40 REM N = NUMERO DE SINAIS
50 READ P, N
60 IF P = 0 THEN IF  N = 0 THEN 600
61 P = P + K
62 IF P< = 0 THEN 66
63 IF P > 40 THEN 66
64 IF P + N > 40 THEN 68
65 GOTO 70
66 PRINT
67 GOTO 50
68 N = 40 - P
70 PRINT TAB (P);
80 FOR J = 1 TO N
90 PRINT "  *",
100 NEXT J
110 PRINT "ƀ"
120 K = K1 + K3
125 K3 = K3 + K2
130 GOTO 50
400 DATA 23, 1, 22, 2, 21, 8, 19, 11, 17, 15, 13, 22, 14, 28, 23, 28
410 DATA 11, 26, 23, 26, 11, 19, 13, 19, 12, 19, 14, 21, 13, 19, 15, 18
440 DATA 16, 20, 16, 18, 17. 18, 16, 19, 16, 16, 16, 14, 15, 9, 15, 7
460 DATA 14, 7, 14, 6, 15, 6, 16, 6, 17, 19, 18, 13, 30, 2, 32, 1, 0, 0, 0
600 END
```

Resultado:

DESLOCAMENTO À DIREITA : K1 = 10
ROTAÇÃO GRADUAL À ESQUERDA: K2 = - 1
LIMITE À DIREITA: POSIÇÃO 40

```
RUN
? 10, -1
                                *
                               **
                             ********
                          ***********
                       ***************
                   *********************
                   *********************
                 ***********************
              **************************
               ****************
            *******************
             *******************
            *******************
             *********************
           *******************
            ******************
            ********************
           ******************
           ******************
         *******************
        ****************
       **************
     *********
    *******
  *******
******
******
******
*******************
*************
           **
            *
READY
```

16.5 GRÁFICO USANDO FUNÇÃO **PLOT X,Y OU SET(X,Y)**

(Adaptado de: Microcomputer graphics — de D. Hearn e M. P. Baker, Prentice-Hall, USA, 1983).

A maioria dos microcomputadores possuem funções próprias para traçar gráficos sobre a tela do vídeo de TV.

A tela é considerada como uma matriz (X,Y) de pontos, cuja dimensão varia de microcomputador para microcomputador.

A função **PLOT X,Y** ou **SET(X,Y)** que são equivalentes, servem para colocar um PONTO (ou uma MARCA) na posição (X,Y) dessa matriz.

As principais funções usadas para comandar o gráfico nessa matriz são:

a) FIXAR MODELO GRÁFICO: **GR** ou **GRAPHICS** ou **GRAPH**
b) LIMPAR A TELA: **CLS** ou **CLEAR**
c) COLOCAR PONTO NA POSIÇÃO (X,Y): **PLOT X,Y** ou **SET(X,Y)** ou **PSET(X,Y)**
d) APAGAR PONTO DA POSIÇÃO (X,Y): **POINTOFF(X,Y)** ou **RESET(X,Y)** ou **PRESET(X,Y)**

Infelizmente não existe uniformidade no nome e na forma dessas funções, e o usuário deve consultar o manual do seu microcomputador para usar a forma correta das mesmas. O MODO GRÁFICO serve para fixar a saída dos resultados sobre a tela em forma gráfica e o usuário não deve usar, nesse caso, comandos do tipo PRINT ou INPUT que requerem a tela para colocar resultados alfanuméricos. O uso do comando GR de modo gráfico em alguns sistemas é dispensado, pois está incorporado no CLS ou CLEAR.

Exemplo: Traçar P pontos aleatórios ou randômicos na tela.

```
NEW
 10 PRINT "FORNECER VALORES MÁXIMOS DE X E Y"
 20 INPUT XM, YM
 30 PRINT "QUANTOS PONTOS DESEJA?"
 40 INPUT P
 50 PRINT "FORNECER UM VALOR AO ACASO ENTRE ZERO E UM COMO
    ·SEMENTE"
 60 INPUT Z
 70 REM LIMPA A TELA
 80 CLS (ou CLEAR)
 90 REM GERAR P PONTOS
100 FOR K=1 TO P
100 LET X=INT (XM* RND(Z))
120 LET Y=INT(YM* RND(Z))
130 SET(X,Y) (ou PLOT X,Y ou equivalente)
140 NEXT K
150 END
RUN
```

Exemplo: Desenhar usando o teclado uma figura qualquer na tela.

Este programa permite traçar, a partir de um ponto inicial (X,Y):

- um ponto PARA CIMA se apertar a tecla "C";
- um ponto PARA BAIXO se apertar a tecla "B";
- um ponto PARA A DIREITA se apertar a tecla "D";
- um ponto PARA A ESQUERDA se apertar a tecla "E"; e
- TERMINAR a figura se apertar a tecla "F".

```
NEW
 10 PRINT "FORNECER VALORES MÁXIMOS DE X E Y"
 20 INPUT XM, YM
 30 PRINT
 40 PRINT "FORNECER COORDENADAS INICIAIS X E Y"
 50 INPUT X, Y
 60 REM MODO GRÁFICO E LIMPAR A TELA
 70 CLS (ou CLEAR)
 80 IF X < 0  OR X > XM  OR Y < O  OR Y > YM  THEN 270
 90 SET (X,Y) (ou PLOT X,Y)
100 REM APERTAR UMA DAS LETRAS PERMITIDAS NO TECLADO"
110 S$=INKEY$
130 IF S$="B"  THEN GO TO 190
140 IF S$="C"  THEN GO TO 210
150 IF S$="D"  THEN GO TO 230
160 IF S$="E"  THEN GO TO 250
170 IF S$="F"  THEN GO TO 300
180 GO TO 80
190 Y=Y+1
200 GO TO 80
210 Y=Y−1
220 GO TO 80
230 X=X+1
240 GO TO 80
250 X=X−1
260 GO TO 80
270 PRINT "PONTO FORA DO LIMITE PERMITIDO"
300 END
```

Nota: O comando de controle INKEY$ permite receber um sinal alfa-numérico apertando qualquer tecla do teclado.

16.6 EXERCÍCIOS DE APLICAÇÃO

1. Traçar, usando o levantamento ponto a ponto do item 16.1, a imagem de qualquer outra figura. Por exemplo: um quadrado, um losango, uma flor, perfil de um rosto, de um automóvel etc.

2. Introduzir, nos exemplos dados, outros tipos de transformação de uma figura. Por exemplo: operação de aumento e diminuição de escala.

3. Escrever um programa Basic que gire a figura do item 16.2 em 180 graus em torno de um eixo vertical centralizado, isto é, de modo que a figura do item 16.2 seja rebatida, mudando o lado direito e o lado esquerdo.

Aplicação: Sistema de Equações Lineares 17

17.1 CONCEITO BÁSICO E REGRA DE CRAMER

Um sistema de equações é dado na forma:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= c_1 \\
a_{21}x_1 + \qquad\qquad + a_{2n}x_n &= c_2 \\
a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= c_n
\end{aligned}
$$

A regra de Cramer para resolver um sistema de duas equações é:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 &= c_1 \\
a_{21}x_1 + a_{22}x_2 &= c_2
\end{aligned}
$$

$$
x_1 = \frac{\begin{vmatrix} c_1 & a_{12} \\ c_2 & a_{22} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}} \quad \text{e} \quad x_2 = \frac{\begin{vmatrix} a_{11} & c_1 \\ a_{21} & c_2 \end{vmatrix}}{\begin{vmatrix} a_{11} & c_{12} \\ a_{21} & a_{22} \end{vmatrix}}
$$

Para um sistema com mais de três equações o cálculo dos determinantes torna-se trabalhoso e devemos procurar outro método de resolução.

17.2 MÉTODOS ITERATIVOS

O sistema de equações:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = c_1$$

$$\ldots\ldots$$

$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = c_n$$

é rearranjado, de modo a isolar x_1 na 1ª equação, x_2 na 2ª equação etc.

$$x_1 = \frac{1}{a_{11}}(c_1 - a_{12}x_2 - a_{13}x_3 - \ldots - a_{1n}x_n)$$

$$\ldots\ldots$$

$$x_n = \frac{1}{a_{nn}}(c_n - a_{n1}x_1 - a_{n2}x_2 - \ldots - a_{n,n-1}x_{n-1})$$

O processo iterativo assume uma solução inicial $(x_1^{(0)}, \ldots, x_n^{(0)})$ que, substituídos nos lados direitos das equações rearranjadas, fornece a próxima solução $(x_1^{(1)}, \ldots, x_n^{(1)})$ e assim por diante.

A iteração termina, por exemplo, quando, entre 2 soluções sucessivas, temos:

$$\left|\frac{x_1^{(i+1)} - x_1^{(i)}}{x_1^{(i+1)}}\right| + \left|\right| + \ldots + \left|\frac{x_n^{(i+1)} - x_n^{(i)}}{x_n^{(i)}}\right| \leqslant \text{precisão dada}$$

Exemplo:

Seja o sistema:

$$3x_1 + x_2 + x_3 = 10$$
$$x_1 + 5x_2 + 2x_3 = 21$$
$$x_1 + 2x_2 + 5x_3 = 30$$

As fórmulas de iteração serão:

$$x_1^{(i+1)} = (1/3)\ (10 - x_2^{(i)} - x_3^{(i)})$$
$$x_2^{(i+1)} = (1/5)\ (21 - x_1^{(i)} - 2x_3^{(i)})$$
$$x_3^{(i+1)} = (1/5)\ (30 - x_1^{(i)} - 2x_2^{(i)})$$

Usando a solução inicial: $x_1 = c_1/a_{11} = 10/3$,

$x_2 = c_2/a_{22} = 21/5$

e $x_3 = c_3/a_{33} = 30/5$

temos os seguintes valores (a solução inicial $x_1 = x_2 = x_3 = 0$ resultará após uma iteração, na solução inicial acima).

iteração	1	2	16	17	18
x_1	3,333	-0,076 . . .	0,998	1,001	0,999
x_2	4,200	1,111 . . .	1,998	2,001	2,000
x_3	6,000	3,654 . . .	4,998	5,001	5,000

17.3 MÉTODO DE ITERAÇÃO DE GAUSS–SEIDEL

Parece ser vantajoso usar cada novo valor x_i obtido em uma iteração imediatamente no cálculo dos demais valores durante a mesma fase da iteração. É o que faz o método de Gauss-Seidel que usa as fórmulas: (o valor $x_1^{(i+1)}$ é imediatamente usado para obter $x_2^{(i+1)}$, etc.)

$$x_1^{(i+1)} = (1/a_{11})(c_1 - a_{12}x_2^{(i)} - \ldots - a_{1n}x_n^{(i)})$$
$$x_2^{(i+1)} = (1/a_{22})(c_2 - a_{12}x_1^{(i+1)} - a_{23}x_3^{(i)} \ldots - a_{2n}x_n^{(i)})$$
$$x_n^{(i+1)} = (1/a_{nn})(c_n - a_{n1}x_1^{(i+1)} - a_{n2}x_2^{(i+1)} - \ldots -$$
$$-a_{n,n-1}x_{n-1}^{(i+1)})$$

O exemplo do exercício anterior converge mais rapidamente por este método.

iteração	1	2	5	6	7
x_1	3,333	-0,067	1,006	1,001	1,000
x_2	4,200	1,813	1,996	2,000	2,000
x_3	6,000	5,288	5,006	5,000	5,000

17.4 SISTEMA DE TRÊS EQUAÇÕES PELO MÉTODO ITERATIVO DE GAUSS-SEIDEL

Resolver por Basic o mesmo sistema estudado:

$$3x_1 + x_2 + x_3 = 10$$
$$x_1 + 5x_2 + 2x_3 = 21$$
$$x_1 + 2x_2 + 5x_3 = 30$$

```
100 DIM A (3, 3), B (3)
105 PRINT "FORNECER COEF A (I, J)"
110 FOR I = 1 TO 3
120 FOR J = 1 TO 3
130 READ A (I, J)
140 NEXT J
150 NEXT I
155 PRINT "COEF B (I)"
160 FOR I = 1 TO 3
170 READ B (I)
180 NEXT I
190 PRINT "VALORES INICIAIS X1, X2, X3"
200 INPUT X1, X2, X3
210 FOR L = 1 TO 10
220 LET X1 = 1/A (1,1) * (B (1) - A (1, 2) *X2 - A (1,3) * X3)
230 LET X2 = 1/A (2, 2) * (B (2) - A (2, 1) * X1 - A (2, 3) * X3)
240 LET X3 = 1/A (3, 3) * (B (3) - A (3, 1) * X1 - A (3, 2) * X2)
250 PRINT "ITERACAO", L, X1, X2, X3
260 NEXT L
270 DATA 3, 1, 1, 1, 5, 2, 1, 2, 5
280 DATA 10, 21, 30
290 END
```

```
RUN
FORNECER COEF A (I, J)                  (lidos no READ-DATA)
COEF B (1)
VALORES INICIAIS X1, X2, X3
?3.333, 4.2, 6
ITERACAO 1              3.333    4.2      6
ITERACAO 2             -0.067    1.813    5.288


ITERACAO 7              1.000    2.000    5.000
. . .
ITERACAO 10             1.000    2.000    5.000
READY
```

17.5 EXERCÍCIOS DE APLICAÇÃO

1. Escrever um programa em Basic que calcule o determinante de uma matriz 3x3 e então resolva um sistema de 3 equações lineares a 3 incógnitas (Regra de Cramer).

2. Altere o programa de iteração pelo método Gauss-Seidel dado no capítulo, para o método iterativo comum. Qual a diferença entre os dois métodos?

3. Resolver os seguintes sistemas lineares:

   a) $2x + 3y = 3$
      $5x - 4y = 14$

   b) $4x + 2y - 6z = 10$
      $9x - 6y + 3z = 15$
      $5x - 3y - z = 16$

   Escrever um programa Basic, usando a Regra de Cramer.

4. Resolver o seguinte sistema de equações, através de um programa BASIC, com método de Gauss-Seidel:

   $x + 2y + 3z = c$
   $3x + 10y - 4z = d$
   $-2x - 4y + z = e$

para os seguintes valores de c, d, e:

c =	6	6	14	-6	4
d =	-29	9	11	-9	-1
e =	9	-5	-7	5	-1

5. Uma fábrica produz 3 tipos diferentes de produtos que são processados em 3 departamentos diferentes. A tabela de gasto de horas de cada Departamento por cada produto, bem como o número total disponível de horas de cada departamento é a seguinte:

	Consumo de horas (do Departamento)		
	A	B	C
Produto 1	3	1	1
Produto 2	1	5	2
Produto 3	1	2	5
Total de horas Disponíveis	10	21	30 horas

Quantas unidades de cada produto é possível fabricar, utilizando-se plenamente a capacidade disponível em cada departamento? Formular o problema como sistema de equações lineares e resolver pelo método de Gauss-Seidel.

6. Em um circuito elétrico de 6 nós, as resistências estão em ohms e o potencial aplicado entre A e B é de 100 volts.

As equações dadas pela Lei de Kirchoff nos 6 nós, em função das potências v_i no nó i, são:

Nó 1:	$11 v1 - 5 v2 - v6 = 500$
Nó 2:	$-20 v1 + 41 v2 - 15 v3 - 6 v5 = 0$
Nó 3:	$- 3 v2 + 7 v3 - 4v4 = 0$
Nó 4:	$- v3 + 2 v4 - v5 = 0$
Nó 5:	$- 3 v2 - 10 v4 + 28 v5 - 15 v6 = 0$
Nó 6:	$- 2 v1 - 15 v5 + 47 v6 = 0.$

Resolver o sistema pelo método de Glauss-Seidel para as dez primeiras iterações e usar como solução incial v1 = v2 = v3 = v4 = v5 = v6 = 1.

Escrever um programa em Basic e testar o programa manualmente para 3 iterações.

Aplicação: Integração Numérica 18

18.1 CONCEITO BÁSICO

Freqüentemente não é possível obter diretamente o valor da integral de uma função, ou porque f(x) é difícil de ser integrada ou porque f(x) é fornecida em forma de tabela, ou valores experimentais obtidos por medição.

O procedimento adotado em tais casos é substituir a função f(x) por um polinômio de interpolação p(x) e efetuar numericamente o cálculo da integral de p(x).

18.2 REGRA DO TRAPÉZIO

O processo mais simples de obter o valor aproximado de uma integral é determinar as áreas dos polígonos ou trapézios formados, ligando-se dois pontos consecutivos de f(x) por uma reta (e temos uma interpolação linear).

Por exemplo, os pontos x_1 e $x_2 = x_1 + h$, separados pelo intervalo h, fornecem área: (Fig. 18.1.).

$$\text{Área} = \frac{f(x_1) + f(x_1 + h)}{2} . h$$

Para n pontos $f(x_1), f(x_2), \ldots, f(x_n)$ basta somar as áreas dos trapézios obtidos e temos:

$$\int_{x_1}^{x_n} f(x)dx = \frac{h}{2} \left[f(x_1) + 2\, f(x_2) + \ldots + 2\, f(x_{n-1}) + f(x_n) \right]$$

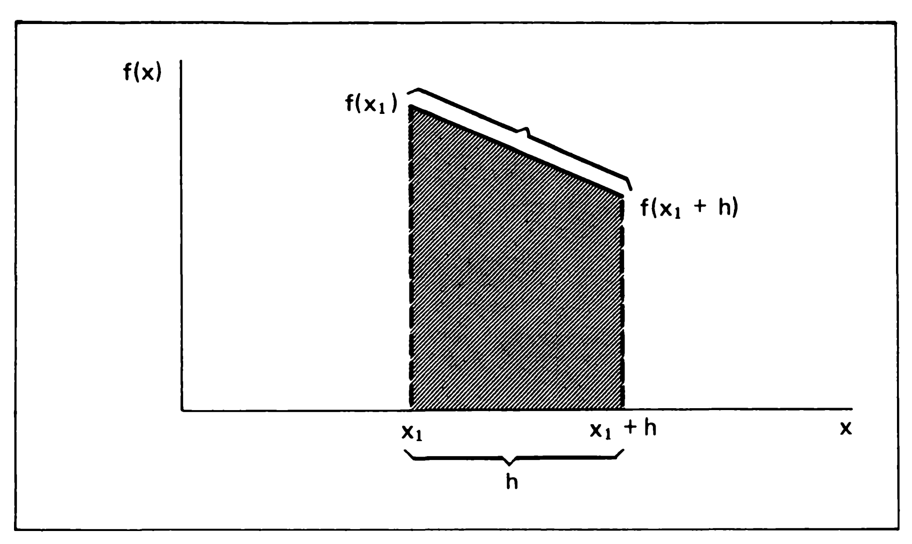


Figura 18.1. *Integração pela regra do trapézio.*

18.2.1 Aplicação

Obtenção da distância percorrida por um veículo espacial

Um veículo espacial foi lançado da plataforma no instante t = 0, e é possível obter a certos intervalos de tempo o registro de sua velocidade que é variável. No instante t qualquer é possível obter a distância total x(t) percorrida pelo veículo espacial, através da integral:

$$x(t) = \int_0^t v(t)\,dt \qquad \text{para } x(0) = 0.$$

A obtenção da distância percorrida x(t) só é possível através da integração numérica, pois desconhecemos como a velocidade v(t) varia.

Por exemplo, se o registro de dados obtidos foi:

T (seg.)	v(t) km/seg
0	1,00
2	1,82
4	2,09
6	3,52
8	4,75
10	6,81

Aplicando a fórmula vista, com h = 2:

$$x(t) = \int_0^t v(t)\,dt =$$

$$= (2/2)\ (1{,}00 + 2 \times (1{,}82 + 2.09 + 3{,}52 + 4.75) + 6{,}81)$$

$$= 32.17 \text{ km}.$$

Quando um motor funciona durante t segundos a energia total consumida é dada por:

$$E(t) = \int_0^t P(t)\,dt.$$

Como na aplicação anterior, só possuímos registro de dados experimentais dos valores da potência P(t) consumida e a energia E (t) só poderá ser obtida através da integração numérica.

Por exemplo, seja um motor elétrico que consumiu energia de acordo com a seguinte tabela:

$P(t) = 500$ watts para $0 \leq t < 1$ hora;

$P(t) = 700$ watts para $1 \leq t < 2$ horas

$P(t) = 1000$ watts para $2 \leq t \leq 4$ horas.

A energia total consumida até o instante M dado será a área que fica sob a curva de valores de P(t) (Fig. 18.2).

- calcular a energia total consumida no instante M e
- custo dessa energia. O custo unitário é Cr$ 5,00 watts/hora.

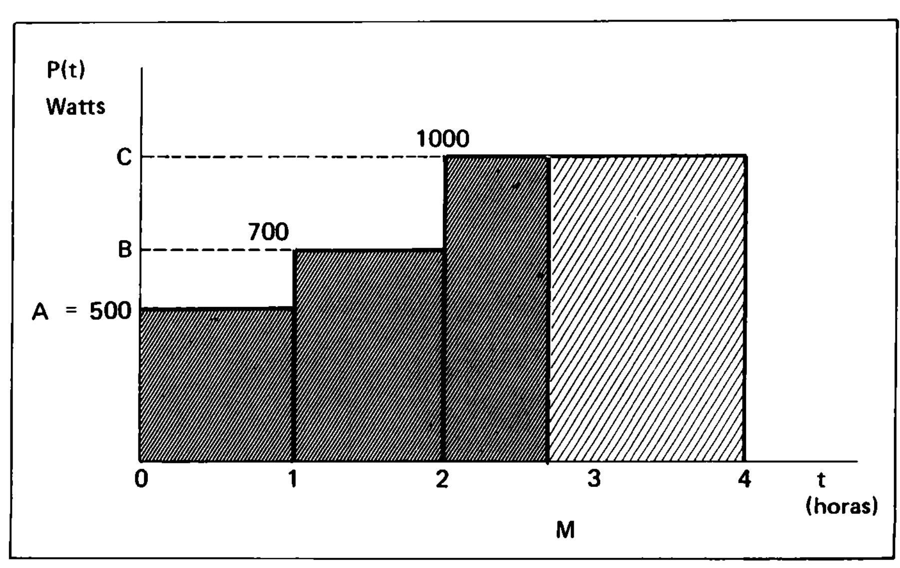


Figura 18.2. *Consumo de energia do motor elétrico.*

Resultado obtido

```
0010 PRINT "A, B, C, H, M"
0020 INPUT A, B, C, H, M
0030 P = 0
0040 I = 0
0050 X = A * 2
0060 Y = 1
0070 GOSUB 1000
0080 X = B * 2
0090 Y = 2
0100 GOSUB 1000
0110 X = C * 2
0120 Y = 4
0130 GOSUB 1000
0140 I = I * H/2
0150 PRINT "ENERGIA TOTAL CONSUMIDA : " ;I; "WH"
0160 J = J * 5 * H/2
0170 PRINT "CUSTO EM DUAS HORAS CR$ " ;J; ",00"
0180 STOP
1000 I = I + X
1001 P = P + H
1010 IF P > M THEN IF P > = 2 + H THEN 1030
1020 J = I
1030 IF P < Y THEN 1000
1040 RETURN
1050 END
```

```
RUN
A, B, C, H, M
? 500, 700, 1000, .5,2
ENERGIA TOTAL CONSUMIDA: 3200 WH
CUSTO EM DUAS HORAS CR$ 6000,00
STOP 0180
READY
```

Valores de
A, B, C, H = intervalo de integração e instante
M = 2 horas

18.3 EXERCÍCIOS DE APLICAÇÃO

1. Escrever um programa em Basic para calcular a integral da função qualquer f(x), dada por n pontos $f(x_1)$, $f(x_2)$, . . ., $f(x_n)$.

   Usar a regra do trapézio.

   Testar o programa com os valores do Exercício do item 18.2.

2. Escrever um programa em Basic para calcular numericamente o valor de:

$$I = \int_{-1}^{+1} (1 - 2x^2 + x^3)\, dx \text{ para incremento } h = 0.1.$$

3. Calcular o valor da área sob a curva f(x) cujos valores foram experimentalmente obtidos:

x	f(x)
0	1,00
0,5	1,70
1,0	2,00
1,5	2,25
2,0	2,50

x	f(x)
2,5	2,40
3,0	1,80
3,5	1,50
4,0	1,40

Apêndice A

Exemplos de sub-rotinas para usar arquivos de discos em microcomputador

A -1) INTRODUÇÃO:

O processamento comercial de serviços rotineiros (por exemplos, número de itens de dados superior a 50 ou 100) requer o uso e a formação de Arquivo de Dados que em microcomputadores quase sempre é feita usando-se disquetes. Infelizmente, os comandos e as rotinas para formar e para usar tais arquivos não são padronizados, surgindo daí uma inevitável dificuldade na apresentação de seu uso em textos didáticos.

Para o uso de arquivo em disquetes, o usuário deve sempre consultar o manual de seu próprio microcomputador.

A própria técnica de formação e uso eficiente de arquivos depende muito do conhecimento das técnicas de organização e acesso aos arquivos de dados. Como fonte de consulta de nível geral sobre organização e acesso de arquivos, recomendamos a leitura de:

Processamento de dados nas empresas de T. Shimizu, Atlas, SP
e *Programação COBOL* de T. Shimizu, Atlas, SP.

Apresentaremos, a seguir, resumos e exemplos de comandos e sub-rotinas para manipular arquivos em diversos microcomputadores. Esses exemplos devem servir apenas para comparação e ilustração dos comandos. O uso exato dos mesmos deve ser procurado nos manuais dos respectivos microcomputadores.

A - 2) BASIC DA APPLE

```
1010 D$ = CHR$ (4)                       (para abrir um arquivo)
1020 PRINT D$; "OPEN", F$
1030 PRINT D$; "READ OR WRITE"; F$
1040 RETURN

1200 PRINT D$; "CLOSE"                   (para fechar um arquivo)
1210 RETURN

3100 INPUT V1, V2                        (ler um item de dado)
3110 INPUT S$
3120 RETURN

4100 PRINT V1; " , ."; V2                (escrever um item de dado)
4110 PRINT S$
4120 RETURN
```

A - 3) BASIC DA IBM

```
1000 OPEN F$ FOR INPUT AS #1             (abrir um arquivo F$ de entrada
1010 RETURN                              no buffer #1)

1050 OPEN F$ FOR OUTPUT AS #2            (abrir arquivo F$ de saída no
1060 RETURN                              buffer #2)

1100 CLOSE                               (fechar arquivo)
1110 RETURN

2100 INPUT #1, V1, V2                    (ler um item de dado)
2120 INPUT #1, S$
2140 RETURN
```

3100 PRINT #2, V1; " , "; V2 (escrever um item de dado)

3120 PRINT #2, S$

3140 RETURN

A - 4) NORTH-STAR BASIC

1520 OPEN #1, "REGISTER" (para abrir arquivo)

1540 CLOSE #1, "REGISTER" (para fechar arquivo)

1400 READ #1 %E * 64,. . .,. . .,. . .,. . ., (para ler item)

1500 WRITE #1%E * 64 ,. . .,. . .,. . .,. . ., (para escrever item)

Apêndice B

Tabela parcial dos caracteres ASCII – Valor decimal X: caractere ASCII

33	1	54	6	75	K
34	”	55	7	76	L
35	*	56	8	77	M
36	$	57	9	78	N
37	%	58	:	79	O
38	&	59	;	80	P
39	’	60	<	81	Q
40	(	61	=	82	R
41	)	62	>	83	S
42	*	63	?	84	T
43	+	64	@	85	U
44	,	65	A	86	V
45	-	66	B	87	W
46	.	67	C	88	X
47	/	68	D	89	Y
48	0	69	E	90	Z
49	1	70	F	91	[
50	2	71	G	92	\
51	3	72	H	93	]
52	4	73	I	94	↑
53	5	74	J	95	←

Apêndice C

Resumo dos principais comandos e funções da linguagem Basic

C - 1) COMANDOS BASIC

Código	Operação	Exemplos
DATA	Contém valores para serem lidos pelo comando **READ** (Cap. 5)	**100 DATA "NOME, 10, 35**
DEF FNa	Define uma função matemática **FNa(X)** que é usada pelo programa (Cap. 9)	**100 DEF FNX(X) = X * *+ 2** **200 DEF FNA(Z)=Z *2 + Z * *2**
DIM	Define o tamanho máximo de variáveis dimensionadas de uma ou duas dimensões (Cap. 7)	**10 DIM X (15), A (30, 2)**
END	Término de execução do programa indicando a mensagem **READY.** Pode ser usado no lugar de **STOP** (Cap. 2)	**90 . . .** **100 END**
FOR. .TO. . . .STEP	Inicia e controla repetição de um trecho de programa que vai até o comando **NEXT** (Cap. 3)	**50 FOR I= TO 5 STEP 0.5** . . . **70 NEXT I**
GOSUB n	Desvio para sub-rotina que começa no comando **n** e retorna após encontrar **RETURN** (Cap. 9)	**1000 GOSUB 300** . . .

Código	Operação	Exemplos
GO TO n ou **GOTO n**	Desvio para o comando de números **n**. Palavra **GOTO** pode ser opcional no comando **IF** (Cap. 4	**70 GO TO 100** **80 IF X = Y THEN 50**
IF...THEN	Executa operação após **THEN** se condição for verdadeira (Cap. 4)	**190 IF X=Y THEN PRINT "IGUAIS"**
INPUT	Lê valores através do teclado do terminal (Cap. 2)	**50 INPUT X, Y, Z** **20 INPUT N$**
LET	Calcula valor de fórmula aritmética Palavra **LET** é opcional (Cap. 2)	**10 LET A = B** **100 Z= (A+B)/(C * 34.0)**
NEXT	Colocado para indicar o fim do trecho de repetição controlado pelo **FOR** (Cap. 3)	**20 FOR I=1 TO 10** **30 PRINT "OBA", I** **40 NEXT I**
ON x	Desvia para o comando **GOTO** ou sub-rotina (**GOSUB**) de ordem k se o valor de x for igual a k (Cap. 9)	**10 ON X GOTO 10, 5, 20** **50 ON A+B GOSUB 300, 50, 400**
PAUSE	Parada temporária do programa podendo recomeçar o processamento com o comando **CONT**. O manual de seu microcomputador deve ser consultado.	**50 PAUSE**
PRINT, **<PRINT** ou **LPRINT**	Escreve comentários e valores na tela do display de TV (**PRINT**) ou na folha de impressão (**<PRINT**) (Cap. 2 e 6)	**30 PRINT A, X, Y, "BOM"** **50 PRINT D$, "DATA"**
PRINT USING	Escreve valores editados e formatados (com sinal **$**, *****, ponto, vírgula) para documentos, relatórios etc. (Cap. 8)	**30 PRINT USING"$$##.##"; B1**

Código	Operação	Exemplos
READ	Lê valores previamente colocados no comando **DATA** (Cap. 5)	**10 READ N$, A1, B2**
REM	Contém comentários sobre o programa. Não é executado. (Cap. 2)	**10 REM INICIO DO CALCULO**
RESTORE	Coloca o ponteiro de indicação dos dados do comando **DATA** para o primeiro valor dos comandos **DATA** (Cap. 5)	**20 READ A, B, C** **30 RESTORE** **40 READ A1, B1, X** **100 DATA 3, 5, 7**
RETURN	Ponto de uma sub-rotina chamada pelo **GOSUB.** (Cap. 9)	**300 . . .** **. . .** **500 RETURN**
STOP	Pára a execução e não recomeça o processamento. Para recomeçar deve-se consultar o manual. (Cap. 1 e 2)	**20 STOP**

C - 1.1) Comandos Basic especiais existentes em outros microcomputadores

IBM-5000	TRS-80	PET
CHAIN	**CMD**	**CLOSE**
OPEN	**ERROR**	**OPEN**
CLOSE	**OUT**	**GET**
DELETE	**RESETE**	**LOAD**
FORM	**RESUME**	**POS**
GET	**SET**	**VERIFY**
PUT		**SPC**
USE		**STEP**
MAT		

C - 2) PRINCIPAIS COMANDOS DE CONTROLE (Cap. 1)

NEW	Limpa memória colocando valor zero
RUN	Executa o programa Basic da memória
LIST	Lista o programa Basic
LIST K1, K2	Lista do comando K1 até K2
SAVE	Copia o programa da memória para outro meio como fita de papel, fita cassete etc.
LOAD	Limpa a memória e carrega um programa (de fita de papel ou cassete) para a memória.
APPEND	Acrescenta o programa da fita de papel ou cassete na memória
CONT	Continua a execução de um programa suspenso pelo comando **PAUSE.**

C - 2.1.) Comandos especiais de controle existentes em alguns microcomputadores.

Motorola M-6800	IBM-5000	TRS-80	PET
EXIT	**ALERT**	**CLEAR**	**SYS**
TRACE-ON	**AUTO**	**DELETE**	**TI$**
TRACE-OFF	**SKIP**	**EDIT**	**WAIT**
CONTROL	**GO**	**TROFF**	
PATCH	**LINK**	**TRON**	
LINE	**MARK**		
DIGITS	**MERGE**		
	PROC		
	RD		
	RENUM		
	UNTIL		

C - 3) PRINCIPAIS FUNÇÕES EM BASIC (Cap. 10)

C - 3.1) Funções Aritméticas

Código	Operação	Exemplos
ABS(X)	–Valor absoluto de X	**Y=ABS (X - Y)**
LOG(X) ou **LN(X)**	–Logaritmo natural de X	**Z=LOG (X+3)**
EXP(X)	–Valor de e^X	**Y=EXP (3.1415)**
SQR(X)	–Raiz quadrada de X	**Z=SQR (2)**
SGN(X)	–Fornece valor 1 se $x > 0$; 0 se $X = 0$ e -1 se $x < 0$	**W = SGN (3 * S1)**
INT(X)	–Maior inteiro contido em X	**Y=INT (X)**
FIX(X)	–Parte inteira do valor X	**Z = FIX (X + 1)**
CDBL(X)	–Converte X para precisão dupla	**Z=CDBL(X)**
CSNG(X)	–Reduz X para precisão simples	**Z=CSNG(X)**
RND(X)	–Fornece valor aleatório entre 0 e 1	**W=RND(0.45)**

C.3.2) Funções Trigonométricas

Código	Operação	Exemplos
SIN(X)	–Seno de X (em radianos)	**Y=SIN(3.14)**
COS(X)	–Co-seno de X "	**Y=COS (Z + 3)**
TAN(X)	– Tangente de X "	**Z=TAN(X)**
ATN(X)	–Arco tangente de X	**Z=ATN(1)**
ACS(X)	–Arco co-seno de X	**Z=ACS(0.5)**
ASN(X)	–Arco seno de X	**Z =ASN(X)**

C.3.3) Funções que manipulam **Strings**

Código	Operação	Exemplos
CHR$(X)	Converte valor X em caractere ASCII	**PRINT CHR$(32)**
ASC$(N$)	Converte primeiro caractere do **STRING N$** em valor numérico do código ASCII	**PRINT ASC$(A$)**
LEFT$(N$,X)	Fornece X caracteres mais à esquerda do **STRING N$**	
RIGHT(N$,X)	Fornece X caracteres mais à direita do **STRING N$**	
MID$(N$,X,Y)	Fornece Y caracteres a partir da posição Y de N$	
STR$(X)	Valor numérico X é tratado como **STRING** de caracteres	**Y$=STR$ (X)**
VAL$(N$)	**STRING** de caracteres **N$** é tratado como valor numérico	**X=VAL ("134")**
_INKEY$	Recebe um caractere do teclado	

C - 3.4) Controle de impressor

TAB(X)	posiciona impressão na coluna X	**PRINT TAB(30); X,Y**

C - 3.5) Funções que permitem acesso à memória

PEEK(X)	Obtém conteúdo da posição endereço X da memória	**PRINT PEEK(1000)**
POKE X, N	Coloca o valor N no endereço X da memória	**POKE 1000, 13**
USR(X)	Permite chamar uma sub-rotina que está escrita em linguagem de máquina	**Y = USR(X)**

Nota: O manual BASIC deve ser consultado para escolha dos valores corretos de X e N.

Respostas e Sugestões aos Exercícios

Capítulo 1

6. b) Após aparecer **READY** na tela, teclar:
   NEW e tecla **RETURN, CR ou ENTER**
   (teclar cada comando Basic) e **RETURN, CR** ou **ENTER** até o último comando **END**.

   d) **LIST NN, NN** ou **LIST NN-NN**

Capítulo 2

1b) Resultados: **TITULO:. . .**

```
?5,10        (valores de X a A fornecidos)
  5.000      10.000          15.000
READY
```

2c)

```
10 INPUT X, Y
20 LET Z = (X * X + Y * Y)/3
30 PRINT Z
40 END
   RUN
```

3. Veja item 2.1, exemplo.
4. Veja item 2.5, exemplo.

Capítulo 3

1a) **X = 1 + 1 +. . .+ 1 = 10**

Y = 1 + 2 + 3 +. . .+ 10

1d) **X = 1, 3, 5, 7, 9**
Imprime: **BASIC**

2a) Idêntico ao valor Y do ex. 1ª com N qualquer.

2c)
```
10 INPUT U, N
20 W1 = U
30 W = U
40 FOR I = 1 TO N
50 W1 = W1 *(U - I)
60 W = W + W1
70 NEXT I
80 PRINT W
90 END
```

Capítulo 4

1a)	**A = 6**	**A = 3**	**A = 5**
	Escreve: **6**	**3**	**TERMINOU**

2a) (Verdadeiro) OR (falso) OR (verdadeiro) Resultado: verdadeiro

2c) NOT (5X(-3)< = 10) ⇒ NOT (verdadeiro) ⇒ falso

3)
```
 10 INPUT X
 20 IF X< 1 THEN 60
 30 IF< =5 THEN 80
 40 LET F = X *(X + 1)
 50 GO TÓ 100
 60 LET F = X + 2
 70 GO TO 100
 80 LET F = X * X + 1
100 PRINT F
120 END
```

4) Veja item 5.3.

Capítulo 5

2a) Ler (Z = 33 e X = 55)
Ler (Y = -1 e W = 43)
Escrever: Y = -1, W = 43, Z = 33, X = 55
Ler (X = 51)
Escrever: Y = -1, W = 43, Z = 33 e X = 51

3) Para os sete alunos, repetir, as operações: (FOR-NEXT)

– Ler número e notas de cada matéria
– Calcular média do aluno
– Acumular soma das notas de cada matéria
– Escrever: número, notas e média do aluno

Após a repetição:

– Calcular e escrever as médias por matéria de toda a classe de sete alunos.

Capítulo 6

1b)	Basic nível II	1983	
	Basic	Basic	1984

2a) Falta um valor no **DATA** para a variável B3.

3b) Alterar o comando: **105 PRINT "=";**

Capítulo 7

1) **MES = (JAN, FEV, MAR, . . ., DEZ)** vetor de 12 elementos

2) Tabela de número do aluno e notas de Física, Matemática, Português e Química do Exercício 3 proposto no Capítulo 5 é um exemplo de matriz de 7 linhas e 5 colunas.

5) Cálculo da diagonal da matriz A(I, J)

```
50 FOR I = 1 TO N
60 LET D = D + A (I, I)
70 NEXT I
```

Capítulo 8

3. Pelo comando **30 PRINT USING A$; A** temos o resultado:

$3, 457.84

6. **LEFT$ (A$, X) = "VALOR b̸"**

ASC$ (A$) = 86 (valor decimal de V no código ACII, Apêndice B)

VAL$ (A$) = deve dar um ERRO, pois o *string* contido em A**$** não pode ser convertido em valor numérico.

Capítulo 9

3a) Para K e X dados não é possível usar **DEF**. Usar **GOSUB.**

3b) **DEF FNX(X) = 2 * X + X * X**

3c) Usar **GOSUB** – Análogo ao Exercício do item 9.4.

Capítulo 10

Todos os exercícios propostos estão baseados nos exercícios resolvidos do capítulo, bastando efetuar pequenas alterações.

Capítulo 11

Capítulo 12

2) Utilizando somente comando **DATA**: ver Exercício 3 proposto no Capítulo 5.

Capítulo 13

1) **10 LET X = 0.75**

15 FOR I = 1 TO 10

(Programa do Jogo de CARA ou COROA)
Se deu **CARA : C = C + 1** (contador de Nº de CARAS)

100 NEXT I

120 PRINT "NUMERO DE CARAS OCORRIDAS:", C

180 END

5) Roteiro: (Sugestão)

a) Gerar um chute: **Y = RND(X)**
Se Y > **0.70**, então **PRINT "CHUTE FOI FORA" e GO TO END**,
Caso contrário, **PRINT "CHUTE VEIO PARA GOL".**

b) Gerar direção do chute que veio para gol: **Z = RND(X)**
Se Z < 0.5, **chute veio na direção 3 ou 4, então PRINT "GOOL"** e **GO TO END**; caso contrário **PRINT "CHUTE NA DIRECAO 1 ou 2"**

c) Gerar defesa do goleiro: **W = RND(X)**
Se W < 0.5, então **PRINT "GOLEIRO DEFENDEU"** e **GO TO END**; caso contrário, **PRINT "GOOL"**

d) **END**: repete cobrança da penalidade.

Capítulo 14

1) Modificar item 14.2 para Subtração e item 14.3 para Divisão.

4) Modificar e adaptar o item 14.4 para as operações solicitadas.

Capítulo 15

4) $x^3 = C$ com C dado e
$f(x) = x^3 - C, f'(x) = 3x^2$

Capítulo 16

2) Redução de escala: no item 16.2, experimente dividir N por um valor constante e use a parte inteira. Por exemplo: **N = INT (N/2)**. Para aumento de escala, experimente o valor **N = INT (2 * N)**. Mesmo raciocínio para redução ou aumento vertical.

Capítulo 17

2. Ver Exemplo do item 17.4

4) Respostas: (x, y, z) = (1, -2, 3); (1, 1, 1); (1, 2, 3); (-1, -1, -1) e (1, 0, 1)

5) Resposta: Produto 1 = 1
Produto 2 = 2 Produto 3 = 5 unidades

Capítulo 18

1. Regra do Trapézio: Ler valores X(I) e F(I), para I = 1, 2, 3, . . ., N.
S = F (1) + F (N) primeiro e último valor
D = 2 x [F (2) +. . .+ F (N - 1)] Soma valores intermediários F(2), . . ., F (N - 1) e multiplica por dois.
Calcular: (H/2) * (S + D)

3) Seguir Exemplo do item 18.2

Referências Bibliográficas

1 — HEARN, D. e BAKER, M.M. *Microcomputer graphics.* USA, Prentice-Hall, 1983.

2 — SHIMIZU, T. *Processamento de dados nas empresas.* São Paulo, Atlas, 1983.

3 — SHIMIZU, T. *Pesquisa operacional: métodos computacionais, modelos e aplicações.* Rio, Ed. Guanabara Dois, 1984.

4 — TRAVERS, K. e outros *Mathematics teaching.* USA, Harper & Row, 1977.

5 — WADSWORTH, N. *Introduction to low resolution graphics.* USA, SCELBI Publications, 1979.

6 — Manuais diversos sobre programação em Basic.

Impresso em offset



Avenida Bogaert, 64
Vila das Mercês São Paulo
Fone: 914-0233
CEP 04298

com filmes fornecidos pelo editor

BASIC

Tamio Shimizu

- Exercícios e problemas resolvidos
- Aplicações comerciais e científicas
- Simulação, jogos e gráficos
- **Basic** no ensino

Este texto vem oferecer ao estudante os subsídios mínimos para acompanhar a evolução da linguagem **Basic** dos microcomputadores de forma didática e acessível. Inicialmente apresenta definições de termos técnicos próprios da linguagem, além dos primeiros comandos básicos. Os capítulos seguintes apresentam os comandos mais avançados, exemplos de aplicações comerciais e científicas, sempre com farta ilustração prática, possibilitando sua utilização em situações reais. Os capítulos finais descrevem as principais funções fornecidas pelo **Basic** em forma de tabela, além de explicações sobre estrutura dos algoritmos, sistema de equações lineares e integração numérica. Exemplos de aplicações e exercícios de fixação permeiam o livro, possibilitando maior interesse e facilitando o entendimento do leitor. Um capítulo foi destinado à programação de ensino através do microcomputador, mostrando sua utilidade tanto em escolas como no treinamento profissional. Em apêndice encontra-se um resumo dos principais comandos e funções da linguagem **Basic.**

NOTA SOBRE O AUTOR

Tamio Shimizu é professor adjunto da Escola Politécnica da USP. É professor da FEI (Faculdade de Engenharia Industrial) e consultor da IPEI-PEI. Fez cursos a nível pós-doutoral nos Estados Unidos no Union College e no Rensselaer Polytechinic Institute. É autor dos livros **Simulação em computador digital** (Blücher), **Programação Cobol, Processamento de dados nas empresas** e **Processamento de dados — conceitos básicos**, do acervo da Atlas.

APLICAÇÃO

Livro-texto para a disciplina **Introdução à Linguagem Basic** dos cursos profissionalizantes. Material básico para cursos intensivos de **Basic** e demais interessados na linguagem dos microcomputadores.

publicação atlas